Phân Tích Chuyên Sâu về Framework AG2 (Phiên bản 0.9)

1. Giới thiệu về Framework AG2

1.1. Mục Đích Cốt Lõi và Tầm Nhìn

AG2 (trước đây gọi là AutoGen) là một framework lập trình mã nguồn mở được thiết kế để xây dựng các tác nhân trí tuệ nhân tạo (AI agent) có khả năng hợp tác với nhau để giải quyết các nhiệm vụ phức tạp.[1] Mục tiêu chính của AG2 là đơn giản hóa việc phát triển và nghiên cứu AI dựa trên tác nhân (agentic AI) [1], tự định vị mình như một lớp nền tảng tương tự PyTorch cho lĩnh vực Học sâu (Deep Learning).[4] Framework này cung cấp một "khung hội thoại đa tác nhân" (multi-agent conversation framework) như một tầng trừu tượng hóa cấp cao để xây dựng các quy trình công việc dựa trên Mô hình Ngôn ngữ Lớn (LLM).[6]

Tầm nhìn của AG2 mở rộng hơn một thư viện đơn thuần, hướng tới việc trở thành một "Hệ điều hành mã nguồn mở cho Tác nhân AI" ("Open-Source Operating System for Agentic AI") [1] và tạo điều kiện cho các tổ chức trở thành "AI-Native" (tổ chức vận hành dựa trên AI).[10] Thuật ngữ "AgentOS" và "AI-Native" [1] cho thấy tham vọng của dự án không chỉ dừng lại ở việc cung cấp công cụ, mà còn muốn đóng vai trò cơ bản hơn trong cách các ứng dụng AI được xây dựng và điều phối. Điều này thiết lập mục tiêu chính của AG2: cho phép giải quyết vấn đề phức tạp thông qua sự hợp tác của nhiều tác nhân AI, đồng thời trừu tượng hóa sự phức tạp trong tương tác của chúng. Tuy nhiên, việc định vị mình một cách tham vọng như vậy, ví dụ như so sánh với PyTorch [4], đặt ra một tiêu chuẩn rất cao về độ ổn định, chất lượng tài liệu, hỗ trợ hệ sinh thái và tính dễ sử dụng. Tình trạng hiện tại của framework (phiên bản 0.9 được đề cập trong truy vấn người dùng và bản phát hành gần đây được ghi nhận trong [1]) cùng với các vấn đề về khả năng truy cập tài liệu được ghi nhận [11] cho thấy có thể có một khoảng cách giữa tầm nhìn này và sự trưởng thành/ổn định thực tế, đặc biệt là về tài nguyên hướng tới người dùng. Việc nhiều trang tài liệu quan trọng không thể truy cập được cho phiên bản 0.9 mâu thuẫn trực tiếp với các yêu cầu đối với một framework nền tảng như vậy, cho thấy rằng mặc dù tham vọng là rõ ràng, việc thực thi (ít nhất là về tài liệu cho v0.9) có thể vẫn đang trong quá trình hoàn thiện.

1.2. Lĩnh Vực Vấn Đề: Điều Phối Hệ Thống Đa Tác Nhân

AG2 giải quyết sự phức tạp của việc xây dựng các ứng dụng AI tinh vi đòi hỏi nhiều tác nhân AI phải làm việc cùng nhau.[1] Nó đơn giản hóa việc điều phối (orchestration), tự động hóa (automation) và tối ưu hóa (optimization) các quy trình công việc LLM phức tạp.[3] Mục tiêu là tối đa hóa hiệu suất của LLM và khắc phục những điểm yếu của chúng thông qua sự hợp tác đa tác nhân.[22] Vấn đề cốt lõi nằm ở khó khăn trong việc phối hợp nhiều thực thể AI chuyên biệt. AG2 cung cấp cơ sở hạ tầng và các mẫu (patterns) như khung hội thoại để quản lý hiệu quả các tương tác này, dù là hoàn toàn tự động hay có sự can thiệp của con người.[1]

1.3. Đối Tượng Người Dùng Chính

Đối tượng người dùng chính của AG2 bao gồm:

- **Nhà phát triển (Developers):** Những người muốn xây dựng ứng dụng AI tận dụng sức mạnh của nhiều tác nhân hợp tác.[1]
- **Nhà nghiên cứu (Researchers):** Các cá nhân hoặc nhóm tham gia vào việc nghiên cứu và phát

triển hệ thống AI dựa trên tác nhân.[1]
- **Tổ chức (Organizations):** Các công ty và tổ chức muốn triển khai các giải pháp AI tiên tiến cho các trường hợp sử dụng khác nhau.[1]
- **Người thực hành AI (AI practitioners):** Được ghi nhận là những người sử dụng rộng rãi framework này.[23]

Điều này nhấn mạnh rằng AG2 phục vụ cả việc phát triển ứng dụng thực tế và nghiên cứu học thuật/công nghiệp trong lĩnh vực AI dựa trên tác nhân đang phát triển nhanh chóng.

1.4. Nguồn Gốc: Fork từ AutoGen

AG2 được xác định rõ ràng là trước đây được biết đến với tên AutoGen.[1] Nó được mô tả là một bản fork (phân nhánh) do cộng đồng điều khiển từ dự án AutoGen gốc.[2] Một tổ chức mới, AG2ai, đã được thành lập để quản lý việc phát triển AG2 và các dự án liên quan với mô hình quản trị mở (open governance).[5] Dự án AutoGen gốc dường như vẫn tiếp tục dưới sự quản lý của Microsoft, có khả năng tập trung vào nghiên cứu hoặc tích hợp với Semantic Kernel.[25] Giấy phép đã thay đổi từ MIT (AutoGen gốc) sang Apache 2.0 cho AG2.[4]

Việc hiểu rõ sự phân tách này là rất quan trọng. AG2 đại diện cho sự tiếp nối và phát triển do cộng đồng dẫn dắt, có khả năng khác biệt so với hướng đi của Microsoft cho dự án AutoGen gốc. Việc quản trị mở [5] và giấy phép Apache 2.0 [1] báo hiệu một cam kết về sự tham gia rộng rãi hơn của cộng đồng. Các bản fork thường xảy ra do tầm nhìn hoặc ưu tiên khác nhau. Sự nhấn mạnh của AG2 vào quản trị mở [5] cho thấy mong muốn kiểm soát cộng đồng rộng rãi hơn so với một dự án do một công ty dẫn dắt. Đồng thời, Microsoft có thể hướng phiên bản của mình tới việc tích hợp chặt chẽ hơn với hệ sinh thái của họ (Azure, Semantic Kernel được đề cập trong [25]). Sự phân kỳ này có nghĩa là AG2 và AutoGen do Microsoft duy trì có thể phát triển khác nhau, ảnh hưởng đến hỗ trợ dài hạn, bộ tính năng và khả năng tích hợp. Do đó, người dùng chọn AG2 cũng đồng nghĩa với việc chọn con đường phát triển do cộng đồng này định hướng.

2. Kiến Trúc và Khái Niệm Cốt Lõi của AG2

2.1. Triết Lý Kiến Trúc: Hợp Tác Dựa Trên Hội Thoại

Triết lý cốt lõi của AG2 là cho phép hoàn thành nhiệm vụ thông qua các cuộc hội thoại giữa nhiều tác nhân.[1] Framework cung cấp một "khung hội thoại đa tác nhân" như một tầng trừu tượng hóa cấp cao.[1] Điều này bao gồm "lập trình hội thoại" (conversation programming) sử dụng cả ngôn ngữ tự nhiên và mã lệnh.[23] Các tác nhân được thiết kế để có thể "hội thoại" (conversable), nghĩa là chúng có thể gửi và nhận tin nhắn để bắt đầu hoặc tiếp tục cuộc hội thoại.[1]

Điều này nhấn mạnh rằng cơ chế chính cho tương tác tác nhân và thực thi quy trình công việc là thông qua việc truyền tin nhắn và đối thoại, mô phỏng sự hợp tác của con người. Framework cung cấp các cấu trúc (như ConversableAgent, các mẫu như GroupChat) để quản lý các cuộc đối thoại này.

2.2. Các Thành Phần Cơ Bản

Các thành phần cốt lõi của AG2 bao gồm:

- **ConversableAgent:** Một lớp cơ sở (base class) chung, cho các tác nhân có khả năng hội thoại.[1] Nó xử lý việc trao đổi tin nhắn và tạo phản hồi.[1] Các tính năng chính bao gồm khả năng tự động trả

lời (auto-reply) [3] và khả năng mở rộng thông qua việc đăng ký các hàm trả lời (reply functions).[3]

- **AssistantAgent:** Một lớp con của ConversableAgent [3], thường được sử dụng cho các nhiệm vụ dựa trên LLM như tạo mã lệnh hoặc giải quyết vấn đề.[1] Nó có một lời nhắc hệ thống (system prompt) mặc định [29] và có thể sử dụng nhiều LLM khác nhau.[1]
- **UserProxyAgent:** Về mặt khái niệm, đây là một tác nhân đại diện (proxy) cho người dùng.[3] Theo mặc định, nó yêu cầu đầu vào từ con người [3] nhưng cũng có thể hoạt động tự động.[1] Quan trọng là, nó thường xử lý việc thực thi mã lệnh.[3] Chế độ đầu vào của con người có thể được cấu hình (ALWAYS, NEVER, TERMINATE).[1]
- **GroupChatManager:** Một tác nhân đặc biệt điều phối các cuộc hội thoại có nhiều hơn hai tác nhân tham gia.[30] Nó chọn người nói tiếp theo và phát tin nhắn.[22]

Các lớp cốt lõi này cung cấp các khối xây dựng cơ bản. ConversableAgent là nền tảng, AssistantAgent tận dụng LLM cho các nhiệm vụ, UserProxyAgent làm cầu nối giữa tương tác người dùng và thực thi mã lệnh, và GroupChatManager mở rộng quy mô hội thoại vượt ra ngoài các cặp tác nhân. Việc dựa vào một lớp cơ sở chung như ConversableAgent [1] thúc đẩy tính linh hoạt và khả năng tùy chỉnh cao.[22] Tuy nhiên, lựa chọn kiến trúc này có khả năng chuyển sự phức tạp sang phía nhà phát triển, người phải định nghĩa cẩn thận hành vi của tác nhân và logic tương tác. Việc cung cấp các mẫu cấu trúc như GroupChat và Swarm [1] có thể được xem là một giàn giáo cần thiết để quản lý sự phức tạp tiềm ẩn này cho các trường hợp sử dụng phổ biến. Nếu không có các tầng trừu tượng hóa cấp cao hơn như các mẫu hội thoại được định nghĩa trước, việc phối hợp nhiều tác nhân tùy chỉnh có thể trở nên rất khó khăn. Do đó, sự tồn tại của GroupChat, Swarm, v.v., có thể là hệ quả trực tiếp của thiết kế ConversableAgent cốt lõi, nhằm mục đích cung cấp tính khả dụng cùng với tính linh hoạt.

2.3. Các Khái Niệm Cốt Lõi

Các khái niệm vận hành và tương tác chính trong AG2 bao gồm:

- **Khung Hội Thoại Đa Tác Nhân (Multi-Agent Conversation Framework):** Khái niệm bao trùm cho phép sự hợp tác của tác nhân thông qua đối thoại.[1]
- **Tích Hợp và Sử Dụng Công Cụ (Tool Integration & Use):** Các tác nhân có thể sử dụng các công cụ bên ngoài, thường được triển khai dưới dạng hàm.[1] Điều này bao gồm các công cụ thực thi mã lệnh [3] và có khả năng là các công cụ tìm kiếm (Arxiv: [48], Web: [28]), truy cập dữ liệu, v.v. AG2 cung cấp một mô-đun Interoperability để kết nối các công cụ từ các framework khác.[42] Kỹ thuật "Dependency injection" (tiêm phụ thuộc) được đề cập như một cách để cung cấp ngữ cảnh/công cụ một cách an toàn.[42]
- **Khả Năng Tương Tác (Interoperability):** AG2 hỗ trợ rõ ràng việc tích hợp các công cụ từ các framework phổ biến khác như LangChain và CrewAI bằng cách sử dụng mô-đun Interoperability.[42] Điều này cho phép tận dụng các hệ sinh thái công cụ hiện có. Việc tích hợp PydanticAI cũng được đề cập.[42] Việc bao gồm rõ ràng một mô-đun Interoperability [42] cho các framework như LangChain và CrewAI cho thấy một cách tiếp cận thực dụng của các nhà phát triển AG2. Họ nhận ra giá trị của các hệ sinh thái công cụ hiện có và ưu tiên cho phép người dùng tận dụng chúng thay vì buộc phải chuyển đổi hoàn toàn hoặc yêu cầu triển khai lại. Điều này làm giảm rào cản áp dụng cho những người dùng đã đầu tư vào các framework khác. Xây dựng một thư viện công cụ toàn diện từ đầu là một công việc lớn. Bằng cách cung cấp các cầu nối chính thức [42] đến các thư viện phổ biến hiện có (LangChain, CrewAI), AG2 cho phép người dùng hưởng lợi ngay lập tức từ các hệ sinh thái đó. Đây là một chiến lược thực tế để nâng cao tiện ích của AG2 một cách nhanh chóng và thu

hút một cơ sở người dùng rộng lớn hơn đã quen thuộc với các công cụ đó.
- **Con Người Trong Vòng Lặp (Human-in-the-Loop):** Framework hỗ trợ các quy trình công việc tích hợp đầu vào và sự giám sát của con người.[1] Điều này chủ yếu được quản lý thông qua UserProxyAgent và thuộc tính human_input_mode của nó.[1]

2.4. Vấn Đề Truy Cập Tài Liệu Kiến Trúc

Trong quá trình nghiên cứu, các trang tài liệu chính mô tả kiến trúc và các khái niệm cơ bản (ví dụ: /user-guide/basic-concepts/) cho phiên bản 0.9 đã không thể truy cập được.[12] Điều này cản trở đáng kể việc hiểu đầy đủ về kiến trúc dự định trực tiếp từ nguồn chính của phiên bản 0.9. Do đó, việc phân tích phải dựa nhiều vào tệp README trên GitHub [1], tài liệu AutoGen cũ hơn [22], thông tin suy luận từ các phần tài liệu có thể truy cập được [7], và các nguồn của bên thứ ba.[3] Việc không thể truy cập tài liệu kiến trúc cốt lõi [12] cho phiên bản được chỉ định (0.9) thể hiện một rủi ro đáng kể cho những người có khả năng áp dụng. Việc phải dựa vào các phiên bản cũ hơn (AutoGen 0.2 - ví dụ: [22]), các diễn giải của bên thứ ba, hoặc suy luận từ mã/ví dụ làm tăng đường cong học tập và tiềm năng hiểu lầm hoặc gặp phải các thay đổi đột phá. Tài liệu chính xác, dễ truy cập là rất quan trọng cho việc áp dụng và sử dụng hiệu quả framework. Việc không thể truy cập các giải thích khái niệm cơ bản [12] buộc người dùng phải ghép nối thông tin từ các nguồn có khả năng lỗi thời hoặc không đầy đủ. Điều này làm tăng thời gian phát triển, gây ra sự không chắc chắn về các phương pháp hay nhất hiện tại, và làm dấy lên lo ngại về chất lượng bảo trì và hỗ trợ của dự án cho phiên bản được ghi nhận.

3. Tính Năng, Khả Năng và Trường Hợp Sử Dụng

3.1. Tổng Quan về Các Tính Năng Chính

AG2 cung cấp một loạt các tính năng cốt lõi để xây dựng và điều phối các hệ thống đa tác nhân:

- **Khung hội thoại đa tác nhân:** Nền tảng cho phép các tác nhân giao tiếp và hợp tác.[1]
- **Xây dựng ứng dụng đa dạng:** Cho phép tạo ra các hệ thống hoạt động cho nhiều ứng dụng thuộc các lĩnh vực và độ phức tạp khác nhau.[7]
- **Suy luận và tối ưu hóa LLM nâng cao:** Hỗ trợ các API suy luận LLM nâng cao để cải thiện hiệu suất và giảm chi phí.[5]
- **Sử dụng công cụ:** Tác nhân có thể gọi và thực thi các công cụ bên ngoài.[1]
- **Quy trình công việc tự động & có sự tham gia của con người:** Hỗ trợ cả tương tác tác nhân hoàn toàn tự động và các quy trình tích hợp đầu vào của con người.[1]
- **Các mẫu hội thoại đa dạng:** Cung cấp các mẫu tích hợp sẵn như Two-Agent Chat, Sequential Chat, Group Chat, Nested Chat và Swarm.[1]
- **Thực thi mã lệnh:** Khả năng cho tác nhân tạo và chạy mã lệnh (thường là Python).[1]
- **Truy xuất Thông tin Tăng cường (Retrieval Augmented Generation - RAG):** Cho phép tác nhân truy xuất và sử dụng thông tin từ các nguồn dữ liệu bên ngoài.[1]
- **Tác nhân tùy chỉnh:** Khả năng tùy chỉnh hành vi và khả năng của tác nhân.[5]
- **Hỗ trợ nhiều LLM:** Tích hợp với nhiều mô hình ngôn ngữ lớn khác nhau.[1]

3.2. Các Khả Năng Nâng Cao

3.2.1. Truy xuất Thông tin Tăng cường (RAG)

AG2 tích hợp khả năng RAG, cho phép các tác nhân truy cập và kết hợp kiến thức bên ngoài vào quá

trình suy luận và tạo phản hồi.[1] Điều này rất quan trọng đối với các nhiệm vụ đòi hỏi thông tin cập nhật hoặc kiến thức chuyên ngành cụ thể. Các loại tác nhân cụ thể như RetrieveAssistantAgent và RetrieveUserProxyAgent tồn tại trong các mô-đun contrib để hỗ trợ chức năng này.[30] Các ví dụ minh họa việc tích hợp RAG trong các cuộc trò chuyện nhóm [32] và có khả năng sử dụng cơ sở dữ liệu vector.[28] Cách tiếp cận "agentic RAG" cho thấy việc truy xuất không chỉ là một thao tác tra cứu thụ động mà là một phần tích hợp trong quy trình làm việc của tác nhân, nơi tác nhân có thể lập kế hoạch và thực hiện việc truy xuất như các nhiệm vụ con.[28] Một ví dụ notebook cụ thể (agentchat_groupchat_RAG.ipynb) được cung cấp.[32]

Tuy nhiên, cần lưu ý rằng trang tài liệu 'Next Level Concepts' bao gồm RAG (/user-guide/next-level-concepts/ hoặc /user-guide/advanced-concepts/rag/) đã không thể truy cập được trong quá trình nghiên cứu.[13] Do đó, thông tin chi tiết về cách triển khai và cấu hình RAG trong AG2 v0.9 chủ yếu dựa trên các đề cập trong các phần khác và ví dụ notebook cụ thể.[32]

3.2.2. Thực Thi Mã Lệnh

Một trong những khả năng mạnh mẽ nhất của AG2 là cho phép các tác nhân (thường là UserProxyAgent) thực thi mã lệnh, chủ yếu là Python, được cung cấp trong các tin nhắn.[1] Framework hỗ trợ cả việc thực thi cục bộ (local execution) trên môi trường máy chủ (Shell, Jupyter) và thực thi trong môi trường cô lập thông qua Docker (Shell, Jupyter).[31] Việc thực thi mã lệnh có thể là một phần của cuộc hội thoại giữa một tác nhân viết mã (code writer agent) và một tác nhân thực thi mã (code executor agent).[31] Các ví dụ cụ thể bao gồm các nhiệm vụ như vẽ biểu đồ chứng khoán [3] hoặc phân tích dữ liệu.

Tính năng này cho phép các tác nhân thực hiện các phép tính phức tạp, tương tác với hệ thống bên ngoài, và tạo ra các sản phẩm động (như biểu đồ hoặc tệp dữ liệu). Nó là trung tâm của nhiều trường hợp sử dụng thực tế nhưng đồng thời cũng giới thiệu những cân nhắc bảo mật quan trọng. Nhiều trường hợp sử dụng phức tạp được trích dẫn (vẽ biểu đồ, phân tích dữ liệu, nhiệm vụ mã hóa, tối ưu hóa tiềm năng) phụ thuộc rất nhiều vào tính năng thực thi mã.[3] Điều này cho thấy rằng việc thực thi mã không chỉ là một tiện ích bổ sung mà là một yếu tố cốt lõi cho phép AG2 giải quyết các vấn đề phức tạp, thực tế vượt ra ngoài việc tạo văn bản đơn giản. Các trường hợp sử dụng như tạo biểu đồ [3], thực hiện phân tích tài chính [26], hoặc tối ưu hóa chuỗi cung ứng [46] vốn đòi hỏi tính toán và tương tác với dữ liệu hoặc hệ thống bên ngoài. Việc thực thi mã cung cấp cơ chế để các tác nhân thực hiện các hành động này một cách linh hoạt, làm cho các ứng dụng nâng cao này khả thi trong khuôn khổ AG2.

Mặc dù trang tài liệu 'Next Level Concepts' bao gồm Thực thi Mã lệnh (/user-guide/next-level-concepts/ hoặc /user-guide/advanced-concepts/code-execution/) đã không thể truy cập được [13], một liên kết trực tiếp đến trang thực thi mã của phiên bản latest *đã* có thể truy cập được [31], cung cấp các chi tiết quan trọng. Tài liệu AutoGen cũ hơn cũng đề cập đến nó.[53]

3.2.3. Bảo Mật Thực Thi Mã Lệnh

Việc thực thi mã lệnh do LLM tạo ra đặt ra những rủi ro bảo mật đáng kể cho môi trường máy chủ nơi AG2 đang chạy.[31]

- **Rủi ro Thực thi Cục bộ:** Khi sử dụng trình thực thi mã lệnh cục bộ (LocalCommandLineCodeExecutor), mã được chạy trực tiếp trên cùng nền tảng máy chủ.[31] Điều

này có nghĩa là mã được tạo có khả năng truy cập bất kỳ tài nguyên nào mà người dùng chạy AG2 có quyền truy cập, bao gồm tệp, kết nối mạng và chức năng hệ thống. Điều này tạo ra một lỗ hổng bảo mật đáng kể, vì mã độc hại hoặc được tạo ra kém có thể gây hại cho hệ thống hoặc đánh cắp thông tin nhạy cảm.[31] Tài liệu cảnh báo rõ ràng về việc không nên sử dụng thực thi cục bộ cho môi trường sản xuất vì rủi ro này.[31]

- **Giảm thiểu bằng Docker:** Để giải quyết các rủi ro bảo mật này, AG2 cung cấp tùy chọn sử dụng trình thực thi mã lệnh Docker (DockerCommandLineCodeExecutor).[31] Thực thi Docker chạy mã được tạo bên trong một Docker container cô lập, cung cấp một môi trường sandbox, hạn chế quyền truy cập của mã vào hệ thống máy chủ và tài nguyên của nó.[31] Docker cung cấp sự cô lập, kiểm soát tài nguyên, thực thi dựa trên image và giới hạn quyền truy cập vào hệ thống tệp của máy chủ (chỉ cho phép truy cập vào thư mục làm việc được chỉ định).[31] Kể từ phiên bản AG2 0.2.8, thực thi Docker đã trở thành mặc định.[54]
- **Các Phương pháp Bảo mật Tốt nhất:**
  - Sử dụng Docker cho môi trường sản xuất.[31]
  - Chọn Docker image tối thiểu.[31]
  - Hạn chế đặc quyền của container.[31]
  - Giám sát việc thực thi mã.[31]
  - Sử dụng xác thực người dùng (human_input_mode="ALWAYS") để xem xét mã trước khi thực thi.[31]
  - Cập nhật Docker và image thường xuyên.[31]
  - Dọn dẹp thư mục làm việc tạm thời.[31]
  - Dừng trình thực thi khi không sử dụng.[31]
  - Thực hiện đánh giá mã và quét bảo mật tự động (ví dụ: SAST) vì mã do AI tạo ra thường chứa lỗ hổng.[65]
  - Xem xét sử dụng "dependency injection" để truyền dữ liệu nhạy cảm một cách an toàn hơn.[42]
  - Cân nhắc sử dụng "Tool Calling" (gọi công cụ được định nghĩa trước) thay vì thực thi mã tùy ý như một giải pháp thay thế an toàn hơn.[69]

Sức mạnh của việc thực thi mã [31] trực tiếp tạo ra các rủi ro bảo mật.[31] Mặc dù Docker cung cấp biện pháp giảm thiểu [31], việc bật mặc định [54] và tiềm năng về lỗ hổng trong mã do AI tạo ra [65] tạo ra một sự căng thẳng cố hữu. Các nhà phát triển phải chủ động quản lý rủi ro này, có khả năng hạn chế chức năng (ví dụ: tắt thực thi mã hoặc sử dụng giám sát chặt chẽ hơn của con người) để đảm bảo an ninh. Đề xuất sử dụng Tool Calling bị hạn chế hơn thay thế [69] làm nổi bật sự đánh đổi này. Việc trao cho LLM khả năng thực thi mã tùy ý [31] vốn dĩ nguy hiểm.[31] Mặc dù Docker giúp chứa đựng việc thực thi [31], nó không đảm bảo bản thân mã đó an toàn hoặc không có lỗi.[65] Các nhà phát triển sử dụng AG2 phải liên tục cân nhắc lợi ích của việc thực thi mã tự động so với các hàm ý bảo mật, đòi hỏi cấu hình cẩn thận, giám sát và có khả năng chọn các giải pháp thay thế an toàn hơn như các công cụ được xác định trước.[69]

Bảng sau tóm tắt các tùy chọn thực thi mã lệnh trong AG2 và các cân nhắc bảo mật liên quan:

Phương thức Thực thi	Môi trường	Mức độ Cô lập	Trường hợp Sử dụng Chính	Rủi ro Bảo mật Chính	Giảm thiểu bởi AG2
Cục bộ (Local - Shell)	Hệ điều hành máy chủ	Thấp	Phát triển/Thử nghiệm (Không Sản xuất)	Truy cập tài nguyên máy chủ, thực thi mã độc hại [31]	Không (Yêu cầu chọn không tham gia Docker) [54]
Cục bộ (Local - Jupyter)	Hệ điều hành máy chủ	Thấp	Phát triển/Thử nghiệm (Không Sản xuất)	Truy cập tài nguyên máy chủ, thực thi mã độc hại [31]	Không (Yêu cầu chọn không tham gia Docker) [54]
Docker (Shell)	Docker Container	Cao (Sandbox)	Khuyến nghị cho Sản xuất	Thoát container, lỗ hổng image	Cơ chế thực thi Docker, kiểm soát tài nguyên [31]
Docker (Jupyter)	Docker Container	Cao (Sandbox)	Khuyến nghị cho Sản xuất	Thoát container, lỗ hổng image	Cơ chế thực thi Docker, kiểm soát tài nguyên [31]

3.3. Phân Tích Sâu về Các Mẫu Hội Thoại

AG2 cung cấp các mẫu (patterns) hội thoại tích hợp sẵn để cấu trúc các tương tác đa tác nhân, giúp đơn giản hóa việc triển khai các quy trình công việc phức tạp.[1] Các mẫu chính bao gồm:

- **Two-Agent Chat (Hội thoại Hai Tác nhân):** Mẫu đơn giản nhất, là cuộc hội thoại trực tiếp giữa hai tác nhân.[22] Được khởi tạo thông qua phương thức initiate_chat.[22] Kết quả cuộc trò chuyện có thể được tóm tắt.[33]
- **Sequential Chat (Hội thoại Tuần tự):** Một chuỗi các cuộc hội thoại hai tác nhân, trong đó ngữ cảnh hoặc tóm tắt ("carryover") được chuyển từ cuộc trò chuyện này sang cuộc trò chuyện tiếp theo.[1] Hữu ích cho việc chia nhỏ các nhiệm vụ thành các bước phụ thuộc lẫn nhau.[36] Được khởi tạo thông qua initiate_chats.[35]
- **Group Chat (Hội thoại Nhóm):** Một cuộc hội thoại duy nhất có sự tham gia của nhiều hơn hai tác nhân, được điều phối bởi một GroupChatManager.[1] Người quản lý chọn người nói tiếp theo dựa trên các chiến lược (auto, manual, random, round_robin) hoặc các hàm tùy chỉnh.[22] Việc lựa chọn người nói có thể bị ràng buộc.[36] Có thể được sử dụng để triển khai kiểu điều phối Swarm.[38] Dường như nhạy cảm với chất lượng LLM được sử dụng.[72]
- **Nested Chat (Hội thoại Lồng nhau):** Đóng gói một quy trình công việc đa tác nhân (như một cuộc trò chuyện nhóm) vào bên trong một tác nhân duy nhất, cho phép tái sử dụng nó trong các quy trình công việc lớn hơn.[1] Hữu ích cho tính mô-đun hóa. Ví dụ bao gồm cờ vua hội thoại [22] hoặc xử lý các nhiệm vụ phụ cụ thể như truy xuất đơn hàng.[41]
- **Swarm:** Một mẫu đa tác nhân cụ thể (lấy cảm hứng từ OpenAI Swarm) nơi các tác nhân ủy thác nhiệm vụ cho nhau bằng cách sử dụng các lệnh gọi công cụ HandoffMessage rõ ràng, dựa trên khả năng của chúng.[1] Tất cả các tác nhân chia sẻ cùng một ngữ cảnh tin nhắn.[24] Việc lựa chọn người nói dựa trên thông điệp bàn giao (HandoffMessage) gần đây nhất.[24] Cho phép lập kế hoạch

cục bộ thay vì điều phối tập trung.[24] Yêu cầu các mô hình hỗ trợ gọi công cụ (tool calling).[24] Hỗ trợ chia sẻ biến ngữ cảnh và bàn giao có điều kiện.[41]

Các mẫu này cung cấp cho nhà phát triển các cách khác nhau để cấu trúc sự hợp tác của tác nhân, từ các cặp đơn giản đến các nhóm phức tạp, được định tuyến động. Sự sẵn có của các mẫu hội thoại đa dạng (Two-Agent, Sequential, GroupChat, Nested, Swarm - [1]) phản ánh nhu cầu về các chiến lược phối hợp khác nhau tùy thuộc vào độ phức tạp của nhiệm vụ. Các nhiệm vụ đơn giản có thể sử dụng trò chuyện hai tác nhân, trong khi các vấn đề phức tạp, nhiều mặt được hưởng lợi từ việc điều phối GroupChat hoặc Swarm. Một nhiệm vụ Q&A đơn giản có thể chỉ cần hai tác nhân. Một quy trình nghiên cứu nhiều bước có thể sử dụng Sequential Chat.[36] Một nhiệm vụ thiết kế hợp tác có thể sử dụng GroupChat.[1] Một hệ thống yêu cầu ủy thác nhiệm vụ động dựa trên chuyên môn sẽ tận dụng Swarm.[24] Sự tồn tại của các mẫu đa dạng này cho thấy framework được thiết kế để mở rộng quy mô cơ chế phối hợp của nó theo yêu cầu nhiệm vụ.

Về tài liệu, trang 'Conversation Patterns Deep Dive' có thể truy cập được [36], nhưng nó loại trừ rõ ràng Swarm, hướng người dùng đến nơi khác.[40] Thông tin về Swarm chủ yếu dựa vào tài liệu phát triển của AutoGen [24] và các ví dụ trường hợp sử dụng của AG2.[41] Các trang khái niệm Cơ bản/Nâng cao bao gồm GroupChat/Swarm đã không thể truy cập được.[12]

3.3.1. So Sánh Swarm và GroupChat

Bảng dưới đây so sánh hai mẫu điều phối đa tác nhân chính trong AG2: GroupChat và Swarm.

Đặc điểm chính	GroupChat	Swarm
Phong cách điều phối	Tập trung (Thông qua GroupChatManager) [30]	Phân tán (Thông qua bàn giao giữa các tác nhân) [24]
Lựa chọn Agent tiếp theo	Người quản lý quyết định (LLM/Quy tắc/Người/Ngẫu nhiên) [36]	Dựa trên HandoffMessage gần đây nhất [24]
Luồng điều khiển	Người quản lý điều hướng hội thoại [36]	Tác nhân điều khiển thông qua bàn giao [24]
Chia sẻ ngữ cảnh	Ngữ cảnh được chia sẻ chung [36]	Ngữ cảnh được chia sẻ chung [24]
Nơi lập kế hoạch	Tập trung (Người quản lý) [36]	Cục bộ (Tác nhân quyết định bàn giao) [24]
Cơ chế chính	Chiến lược lựa chọn người nói [36]	Lệnh gọi công cụ HandoffMessage [24]
Trường hợp sử dụng điển hình	Nhiệm vụ hợp tác cần sự phối hợp [1]	Nhiệm vụ cần ủy thác động dựa trên khả năng của tác nhân [24]

3.4. Các Trường Hợp Sử Dụng và Ứng Dụng Minh Họa

Tài liệu chính thức đề cập đến các trường hợp sử dụng như Dịch vụ Khách hàng, Lập kế hoạch Du lịch và Thiết kế Trò chơi.[7] Các ví dụ cụ thể được tìm thấy trong các nguồn khác nhau bao gồm:

- Vẽ biểu đồ chứng khoán [3]
- Giải quyết vấn đề toán học [22]
- Nhiệm vụ lập trình/coding [22]
- Trả lời câu hỏi [23]
- Tối ưu hóa chuỗi cung ứng (OptiGuide) [22]
- Ra quyết định trực tuyến [23]
- Giải trí / Cờ vua hội thoại [22]
- Dịch vụ khách hàng hàng không (ví dụ Swarm/GroupChat) [38]
- Tổng quan tài liệu / Tìm kiếm Arxiv [46]
- Agentic RAG / Trợ lý nghiên cứu [28]
- Thiết kế chip (ví dụ Nvidia) [55]
- Nghiên cứu VC / Phân tích thị trường (ví dụ Better Future Labs) [55]
- Nghiên cứu khoa học / Phân tích dữ liệu vũ trụ học (ví dụ Cambridge) [55]
- Dạy AI kỹ năng mới [38]
- Trực quan hóa dữ liệu [3]
- Học liên tục (Continual Learning) [3]
- Phân tích tài chính [26]
- Tìm kiếm / Duyệt web [28]
- Tạo SQL [46]
- Thu thập dữ liệu web (Web Scraping) [46]

Sự đa dạng của các ví dụ này cho thấy tính linh hoạt của AG2 trên nhiều lĩnh vực khác nhau, từ các nhiệm vụ kỹ thuật như lập trình và phân tích dữ liệu đến các ứng dụng hướng tới kinh doanh hơn như dịch vụ khách hàng và nghiên cứu.

Tuy nhiên, trang Use Cases chính (/use-cases/) và trang chỉ mục Notebook Examples (/use-cases/notebooks/Notebooks) đã không thể truy cập được.[14] Thông tin phải dựa vào các đề cập trong các phần tài liệu khác [7], GitHub [1], tài liệu AutoGen cũ hơn [46], và các nguồn của bên thứ ba.[26] Các liên kết notebook cụ thể như agentchat_swarm_w_groupchat_legacy [38], agentchat_groupchat_RAG [32], tools_interoperability [42], swarm/use-case [47], và agentchat_captainagent_crosstool [44] đã có thể truy cập hoặc được tham chiếu.

4. Cài Đặt và Thiết Lập Ban Đầu

4.1. Yêu Cầu Hệ Thống Tiên Quyết

Để cài đặt và sử dụng AG2, môi trường cần đáp ứng các yêu cầu sau:

- **Python:** Phiên bản Python phải từ 3.9 trở lên và nhỏ hơn 3.14 (>= 3.9, < 3.14).[1]
- **Docker:** Mặc dù không phải là yêu cầu cứng như Python, Docker là cần thiết để sử dụng phương thức thực thi mã lệnh mặc định và được khuyến nghị vì lý do bảo mật.[31] Việc cài đặt Docker trở thành một điều kiện tiên quyết thực tế để tận dụng toàn bộ khả năng của AG2 một cách an toàn. Người dùng không muốn hoặc không thể sử dụng Docker sẽ cần phải tắt tính năng này một cách

rõ ràng và chấp nhận các rủi ro bảo mật liên quan.[54]

- **Hệ điều hành:** Không có yêu cầu hệ điều hành cụ thể nào được đề cập rõ ràng. Tuy nhiên, các ví dụ cung cấp hướng dẫn thiết lập biến môi trường cho cả macOS/Linux và Windows.[81] Người dùng macOS có thể cần sử dụng dấu ngoặc kép khi cài đặt pip với các tùy chọn bổ sung.[81]

4.2. Quy Trình Cài Đặt

AG2 được cài đặt thông qua trình quản lý gói pip của Python. Việc sử dụng môi trường ảo (virtual environment) được khuyến nghị để quản lý các gói phụ thuộc.[82]

- **Lệnh cài đặt cơ bản:** pip install ag2.[4] Các bí danh (alias) pyautogen và autogen cũng có thể được sử dụng và trỏ đến cùng một gói.[4]
- **Cài đặt tối thiểu:** Lệnh cài đặt mặc định chỉ bao gồm các phụ thuộc tối thiểu.[1]
- **Cài đặt với tùy chọn bổ sung (extras):** Để sử dụng các tính năng hoặc tích hợp cụ thể, cần cài đặt các tùy chọn bổ sung. Ví dụ:
  - Tích hợp OpenAI: pip install ag2[openai] [1] (Lưu ý: Kể từ phiên bản 0.8, gói openai không được cài đặt mặc định [81]).
  - Các nhà cung cấp LLM khác: pip install ag2[gemini], pip install ag2[anthropic,cohere,mistral].[81]
  - Tích hợp LangChain: pip install ag2[interop-langchain].[44]
- **Lưu ý cho macOS:** Nếu gặp lỗi "no matches found:", hãy thêm dấu ngoặc kép quanh tên gói, ví dụ: pip install "ag2[openai]".[81]
- **Nâng cấp:** Sử dụng lệnh pip install -U "ag2[extra]" để nâng cấp lên phiên bản mới nhất với các tùy chọn bổ sung mong muốn.[44]

Việc cài đặt tối thiểu mặc định [1] có nghĩa là người dùng *phải* chủ động cài đặt các extras ([openai], [gemini], [interop-langchain], v.v.) [1] cho hầu hết mọi mục đích sử dụng thực tế liên quan đến các LLM cụ thể hoặc tích hợp framework. Điều này thêm một bước thiết lập nhỏ nhưng giữ cho phần phụ thuộc cốt lõi nhẹ nhàng. AG2 nhằm mục đích hỗ trợ nhiều LLM và tích hợp.[1] Việc cài đặt tất cả các phụ thuộc tiềm năng theo mặc định sẽ làm phình to quá trình cài đặt. Yêu cầu các extras rõ ràng cho phép người dùng chỉ cài đặt những gì họ cần, nhưng họ phải biết extras nào tương ứng với chức năng mong muốn của họ.

4.3. Cấu Hình Ban Đầu (API Keys)

Để AG2 tương tác với các Mô hình Ngôn ngữ Lớn, cần phải cấu hình khóa API (API keys):

- **Phương thức cấu hình:**
  - **Biến môi trường:** Đặt khóa API làm biến môi trường hệ thống (ví dụ: OPENAI_API_KEY).[1]
  - **Tệp cấu hình JSON:** Sử dụng tệp JSON, thường được đặt tên là OAI_CONFIG_LIST, để lưu trữ cấu hình cho một hoặc nhiều mô hình.[1] Tệp này cho phép chỉ định tên mô hình (model), khóa API (api_key), và loại API (api_type như "openai", "together").[1]
- **Tải cấu hình:** Sử dụng LLMConfig.from_json(path="OAI_CONFIG_LIST") để tải cấu hình từ tệp JSON.[1]
- **Tệp mẫu:** Một tệp mẫu OAI_CONFIG_LIST_sample được cung cấp để làm khuôn mẫu.[1]

Việc cấu hình chủ yếu liên quan đến việc cung cấp thông tin xác thực cho (các) LLM mong muốn. Tệp OAI_CONFIG_LIST cung cấp một cách có cấu trúc để quản lý nhiều điểm cuối mô hình.

4.4. Vấn Đề Truy Cập Tài Liệu Cài Đặt

Mặc dù có báo cáo về việc không thể truy cập trang Quick Start chính (/quick-start/) trong một lần thử [15], các lần truy cập khác dường như thành công.[7] Hướng dẫn cài đặt cụ thể trong phần Basic Concepts (/user-guide/basic-concepts/installing-ag2/) đã được liên kết [51] và có thể truy cập được.[82] Ngoài ra, hướng dẫn cài đặt cốt lõi cũng có sẵn thông qua tệp README trên GitHub [1] và truy cập Quick Start thay thế.[81] Do đó, việc thiết lập ban đầu dường như được ghi liệu hợp lý mặc dù có các vấn đề rộng hơn trên trang web tài liệu.

5. Sử Dụng AG2: Hướng Dẫn và API

5.1. Hướng Dẫn Bắt Đầu và Ví Dụ

Nhiều tài nguyên có sẵn để giúp người dùng học cách sử dụng AG2:

- **Quick Start Guide:** Cung cấp hướng dẫn cơ bản để thiết lập môi trường và xây dựng quy trình công việc tác nhân đầu tiên.[7] Hướng dẫn này minh họa việc tạo tác nhân (ConversableAgent), cấu hình LLM (LLMConfig), và chạy một cuộc hội thoại đơn giản bằng các phương thức run() và process().[81]
- **Notebook Examples:** Kho lưu trữ GitHub chính chứa một thư mục notebook.[1] Tài liệu cũng liệt kê một phần "Notebook Examples" bao gồm nhiều chủ đề khác nhau.[7] Mặc dù trang chỉ mục chính của các notebook này không thể truy cập được [21], các ví dụ cụ thể vẫn được tham chiếu và có thể truy cập thông qua các liên kết trực tiếp hoặc kho lưu trữ GitHub.[42] Tài liệu AutoGen cũ hơn cũng liệt kê nhiều ví dụ notebook phong phú.[46]
- **Kho lưu trữ build-with-ag2:** Một kho lưu trữ riêng biệt (https://github.com/ag2ai/build-with-ag2) được cung cấp với các ví dụ ứng dụng thực tế đa dạng hơn.[1] Kho lưu trữ này dường như được dự định là nơi trưng bày các triển khai thực tế.
- **Hướng dẫn và Khóa học Bên ngoài:** Các hướng dẫn, bài viết blog và khóa học từ các nguồn bên thứ ba cũng tồn tại, cung cấp thêm tài liệu học tập.[26]

Sự tồn tại của kho lưu trữ build-with-ag2 chuyên dụng [1] và nhiều ví dụ notebook (ngay cả khi trang chỉ mục [21] không thể truy cập, các ví dụ cụ thể được tham chiếu [32]) cho thấy dự án ưu tiên việc học thông qua ứng dụng thực tế. Cách tiếp cận thực hành này có thể hiệu quả nhưng có thể không bù đắp hoàn toàn cho tài liệu khái niệm hoặc API bị thiếu. Cung cấp các ví dụ cụ thể [1] là một cách phổ biến và hữu ích để dạy một framework. Tuy nhiên, việc dựa *chủ yếu* vào các ví dụ, đặc biệt là khi tài liệu cốt lõi bị thiếu [12], có thể dẫn đến sự hiểu biết rời rạc và khó khăn trong việc điều chỉnh các giải pháp cho các vấn đề hơi khác một chút.

5.2. Tổng Quan về API

Mặc dù tài liệu tham khảo API chính thức cho Agent trong phiên bản 0.9 không thể truy cập được [20], các thông tin thu thập được từ các nguồn khác cho thấy cấu trúc API cốt lõi:

- **Các Lớp Chính:**
  - ConversableAgent: Lớp cơ sở cho tất cả các tác nhân có khả năng hội thoại.[1]
  - AssistantAgent: Tác nhân dựa trên LLM cho các nhiệm vụ như giải quyết vấn đề, tạo mã.[1]
  - UserProxyAgent: Proxy cho người dùng, thường xử lý đầu vào/phản hồi của con người và thực thi mã.[1]

    - GroupChat & GroupChatManager: Để quản lý và điều phối hội thoại nhóm.[1]
    - SwarmAgent: Tác nhân được thiết kế cho mẫu Swarm, hỗ trợ bàn giao.[41]
    - LLMConfig: Lớp để cấu hình các mô hình ngôn ngữ lớn.[1]
    - Trình thực thi mã: LocalCommandLineCodeExecutor, DockerCommandLineCodeExecutor.[31]
    - Tác nhân RAG: RetrieveUserProxyAgent (trong contrib).[30]
    - Interoperability: Mô-đun để tích hợp công cụ từ các framework khác.[42]
- **Các Phương Thức Chính (Được đề cập):**
    - initiate_chat(): Bắt đầu cuộc hội thoại hai tác nhân.[22]
    - initiate_chats(): Bắt đầu một chuỗi hội thoại tuần tự.[35]
    - run(): Khởi chạy quy trình công việc tác nhân (trả về iterator).[81]
    - process(): Xử lý iterator từ run() để mô phỏng trải nghiệm trò chuyện.[81]
    - send(): Gửi tin nhắn đến một tác nhân khác.[33]
    - register_reply(): Đăng ký hàm trả lời tùy chỉnh cho tác nhân.[3]
    - generate_reply(): Tạo phản hồi cho tin nhắn đã nhận.[53]
- **Tích hợp bên ngoài:** Google Cloud Vertex AI SDK cung cấp lớp AG2Agent như một trình bao bọc (wrapper) để triển khai các tác nhân AG2.[2]

API dường như tập trung vào các lớp tác nhân và các phương thức tương tác của chúng. Việc thiếu tài liệu tham khảo API chính thức [20] làm hạn chế khả năng cung cấp mô tả chi tiết và có thẩm quyền về API AG2 v0.9. Việc hiểu chữ ký phương thức, tham số, kiểu trả về và phạm vi đầy đủ của các lớp/hàm có sẵn đòi hỏi phải dựa vào các ví dụ mã, tài liệu cũ hơn hoặc các nguồn bên ngoài.

5.3. Vấn Đề Truy Cập Tài Liệu API và Ví Dụ

Như đã đề cập, trang API Reference chính cho Agent (/api-reference/autogen/Agent/) [20] và trang chỉ mục Notebook Examples (/use-cases/notebooks/Notebooks) [21] đều không thể truy cập được trong quá trình nghiên cứu cho phiên bản 0.9.

Điều này không chỉ cản trở việc hiểu sâu về API mà còn làm tăng đáng kể đường cong học tập.[20] Các nhà phát triển có thể gặp khó khăn trong việc hiểu các sắc thái của các tùy chọn cấu hình, hành vi của phương thức và các thành phần có sẵn mà không có tài liệu API toàn diện. Các ví dụ mã cho thấy *cách* sử dụng một số tính năng nhất định, nhưng tài liệu API giải thích *lý do*, các tùy chọn có sẵn và các ràng buộc. Nếu không có tài liệu API dễ truy cập [20], các nhà phát triển phải suy luận chức năng, dựa vào các ví dụ có khả năng lỗi thời hoặc phải đọc mã nguồn, tất cả đều làm tăng thời gian học và nguy cơ lỗi. Hơn nữa, các vấn đề về khả năng truy cập tài liệu [11] cho một phiên bản cụ thể (0.9) có thể là dấu hiệu của sự phát triển liên tục hoặc sự không ổn định trong chính API. Nếu tài liệu không thể được duy trì một cách nhất quán và chính xác cho một bản phát hành được gắn thẻ, điều đó làm dấy lên câu hỏi về sự ổn định của API cơ bản mà nó mô tả. Một bản phát hành ổn định (như v0.9 được đề cập trong [1]) thường ngụ ý một bộ tài liệu ổn định và dễ truy cập tương ứng. Sự không thể truy cập rộng rãi cho thấy hoặc là các vấn đề kỹ thuật với trang tài liệu hoặc là tài liệu (và có thể cả API) vẫn đang thay đổi, ngay cả đối với một bản phát hành được đánh số. Sự không chắc chắn này có thể ngăn cản các nhà phát triển đang tìm kiếm một nền tảng ổn định.

6. Hệ Sinh Thái Cộng Đồng và Hoạt Động

6.1. Phân Tích Hoạt Động Phát Triển (Số liệu GitHub)

Dựa trên ảnh chụp nhanh nghiên cứu (có khả năng dựa trên dấu thời gian "ngày 25 tháng 4 năm 2025" trong [1], mặc dù ngày này có vẻ không chính xác và có thể là lỗi trong quá trình tạo snippet - giả định rằng nó phản ánh trạng thái gần đây), hoạt động trên kho lưu trữ GitHub chính (ag2ai/ag2) cho thấy một dự án đang hoạt động:

- **Commits:** Tổng cộng 4,566 commits.[1] Hoạt động gần đây có thể được xem trong lịch sử commit.[1]
- **Issues:** 208 issues đang mở.[1]
- **Pull Requests (PRs):** 14 PRs đang mở.[1]
- **Releases:** Tổng cộng 42 bản phát hành, bản ổn định mới nhất là v0.9.0.[1]
- **Stars:** 2.4k sao [1] (1 sao trong [4] có thể tham chiếu đến các bản fork).
- **Watchers:** 65 người theo dõi [1] (0 trong [4] có thể tham chiếu đến các bản fork).
- **Forks:** 304 bản fork [1] (0 trong [4] có thể tham chiếu đến các bản fork).

Các số liệu này (đặc biệt là commits, issues, PRs, releases, stars từ [1]) cho thấy một dự án mã nguồn mở đang hoạt động với sự phát triển liên tục và sự quan tâm của cộng đồng. Số lượng issues đang mở cho thấy các lĩnh vực cần cải thiện hoặc sửa lỗi. Khối lượng commits, releases và stars [1] cho thấy hoạt động lành mạnh. Tuy nhiên, số lượng issues đang mở (208 trong [1]) so với số PR đang mở (14 trong [1]) có thể cho thấy rằng việc báo cáo vấn đề đang vượt quá tốc độ giải quyết, có khả năng chỉ ra một cộng đồng hoặc nhóm bảo trì đang hoạt động nhưng có thể bị hạn chế về nguồn lực. Mặc dù hoạt động là có, khả năng quản lý các vấn đề đến có thể là một điểm nghẽn.

6.2. Hỗ Trợ Cộng Đồng và Đóng Góp

AG2 được duy trì bởi các tình nguyện viên từ nhiều tổ chức khác nhau [1] và tích cực tìm kiếm sự tham gia của cộng đồng:

- **Hướng dẫn Đóng góp:** Cung cấp một Hướng dẫn Đóng góp (Contributor Guide) chi tiết.[8]
- **Cách thức Đóng góp:** Khuyến khích đóng góp thông qua Pull Requests (PRs), đánh giá PR, giúp đỡ trên Discord, cải thiện tài liệu, thêm ví dụ, báo cáo lỗi và yêu cầu tính năng.[84]
- **Hỗ trợ Người mới:** Cung cấp các "good first issues" (vấn đề tốt cho người mới bắt đầu).[84]
- **Đóng góp Nâng cao:** Có các "roadmap issues" (vấn đề liên quan đến lộ trình) cho những người đóng góp có kinh nghiệm.[84]
- **Tương tác Cộng đồng:** Tổ chức các cuộc họp scrum cộng đồng và thảo luận kỹ thuật thường xuyên.[84] Có một máy chủ Discord [5] và các cuộc thảo luận trên Reddit so sánh AG2/AutoGen với các framework khác.[27]
- **Ghi nhận:** Có chương trình ghi nhận người đóng góp [84] và một cuộc khảo sát dành cho người đóng góp.[84]

Sự hiện diện của Hướng dẫn Đóng góp chi tiết [84], các "good first issues," và các cuộc họp cộng đồng [84] thể hiện một nỗ lực có ý thức để giảm bớt rào cản cho những người đóng góp mới và tích hợp họ vào quy trình phát triển. Điều này rất quan trọng đối với sức khỏe và sự bền vững lâu dài của một bản fork do cộng đồng điều khiển. Các dự án mã nguồn mở phát triển mạnh nhờ sự đóng góp. Việc cung cấp hướng dẫn rõ ràng [84], các điểm vào dễ dàng ("good first issues"), và các cơ hội tương tác thường xuyên (họp) giúp những người đóng góp tiềm năng dễ dàng tham gia và trở thành thành viên tích cực của cộng đồng, điều này rất quan trọng đối với một dự án dựa vào nỗ lực tình nguyện.[1]

6.3. Giấy Phép

AG2 sử dụng giấy phép Apache 2.0.[1] Đây là một sự thay đổi so với giấy phép MIT của AutoGen gốc.[4] Apache 2.0 là một giấy phép mã nguồn mở cho phép, thường được coi là thân thiện với doanh nghiệp, cho phép sử dụng, sửa đổi và phân phối với các điều khoản cấp bằng sáng chế. Sự thay đổi này đáng chú ý. Mặc dù cả hai đều là giấy phép cho phép, Apache 2.0 bao gồm một điều khoản cấp bằng sáng chế rõ ràng và yêu cầu nêu rõ các thay đổi, điều này có thể là những cân nhắc quan trọng đối với việc áp dụng và đóng góp của doanh nghiệp. Lựa chọn này phù hợp với việc chuyển sang mô hình quản trị mở [5] và có thể phản ánh ý định thu hút sự tham gia và đóng góp rộng rãi hơn từ ngành công nghiệp vào bản fork AG2 dưới cấu trúc quản trị mới của nó.

7. Phân Tích So Sánh: AG2 và Các Giải Pháp Thay Thế

7.1. Các Giải Pháp Thay Thế Được Xác Định

AG2 tồn tại trong một hệ sinh thái đang phát triển gồm các công cụ xây dựng tác nhân AI và điều phối quy trình công việc LLM. Các giải pháp thay thế hoặc framework tương đương được đề cập bao gồm:

- **LangChain:** [6]
- **LangGraph:** [6]
- **CrewAI:** [42]
- **LlamaIndex:** [6]
- **Google's Agent Development Kit (ADK):** [6]
- **Phidata:** [85]
- **Pydantic AI:** [42]
- **Smol Agents:** [27]
- **Storm:** [27]
- **Semantic Kernel:** (Hướng đi tiềm năng của Microsoft cho AutoGen) [25]
- **Vertex AI Agent Engine:** (Nền tảng triển khai hỗ trợ AG2, LangChain, v.v.) [2]
- Các nền tảng/framework Tác nhân AI nói chung.[91]

7.2. Thảo Luận So Sánh

Dựa trên thông tin hạn chế có sẵn:

- **AG2 vs. LangChain/LangGraph:** AG2 cung cấp một khung hội thoại đa tác nhân như một tầng trừu tượng hóa cấp cao.[6] LangChain được cho là dễ dàng hơn cho các trường hợp sử dụng cơ bản do cấu hình được xác định trước [6] nhưng có thể có đường cong học tập dốc hơn cho người dùng không chuyên về kỹ thuật so với các công cụ không cần mã.[88] LangGraph cung cấp cách tiếp cận dựa trên đồ thị với khả năng human-in-the-loop/replay nâng cao.[6] Một bình luận của người dùng cho rằng AG2/AutoGen có đường cong học tập dễ dàng hơn nhưng tính linh hoạt/khả năng mở rộng kém hơn so với LangGraph.[86] AG2 hỗ trợ rõ ràng việc tích hợp các công cụ LangChain.[42] Vertex AI Agent Engine hỗ trợ triển khai các tác nhân được xây dựng bằng AG2, LangChain và LangGraph.[6] Công cụ Pandas Dataframe của LangChain được lưu ý là có rủi ro bảo mật tương tự như thực thi mã nói chung.[62]
- **AG2 vs. CrewAI:** CrewAI được đề cập là tập trung vào sự hợp tác liền mạch giữa con người và AI [43], trong khi AG2 tập trung vào việc tự động hóa điều phối đa tác nhân.[43] CrewAI được xây dựng trên LangChain.[62] AG2 hỗ trợ tích hợp các công cụ CrewAI.[42] CrewAI cho phép các tác nhân ủy

thác công việc.[62] Một số nhà phát triển chọn CrewAI do quen thuộc với LangChain, trong khi những người khác thấy AG2 tùy biến hơn.[62] Nhiều người coi chúng thực hiện các nhiệm vụ tương tự.[62]

- **Điểm mạnh của AG2 (Ngụ ý/Nêu rõ):** Tập trung vào *hội thoại* đa tác nhân làm mô hình cốt lõi [1], các mẫu tích hợp sẵn như Swarm và GroupChat để điều phối [1], tích hợp mạnh mẽ việc thực thi mã lệnh [31], các tác nhân có thể tùy chỉnh/hội thoại [22], tính linh hoạt của human-in-the-loop.[1]
- **Điểm yếu của AG2 (Ngụ ý/Nêu rõ):** Các vấn đề tiềm ẩn về khả năng mở rộng/linh hoạt so với các phương pháp dựa trên đồ thị như LangGraph (bình luận người dùng [86]), các vấn đề về khả năng truy cập tài liệu [11], rủi ro bảo mật liên quan đến thực thi mã cần quản lý cẩn thận.[31] Sự nhạy cảm của GroupChat đối với chất lượng LLM.[72]

So sánh cho thấy vị trí créneau của AG2 là sự tập trung vào các mô hình hội thoại và các mẫu điều phối tích hợp sẵn cho nhiều tác nhân. Các framework như LangGraph cung cấp quản lý trạng thái/đồ thị rõ ràng hơn, có khả năng mang lại sự linh hoạt hơn cho các quy trình công việc phức tạp, phi tuyến tính nhưng có thể với đường cong học tập dốc hơn. CrewAI có vẻ tương tự nhưng có thể nhấn mạnh khía cạnh hợp tác người-AI khác đi. Khả năng tương tác của AG2 [42] là một lợi thế thực tế.

Không gian framework tác nhân AI đang chuyên môn hóa. AG2 tập trung vào điều phối đa tác nhân lấy hội thoại làm trung tâm.[1] LangGraph tập trung vào các quy trình công việc dựa trên đồ thị, có trạng thái.[6] LangChain cung cấp các thành phần ứng dụng LLM rộng hơn.[6] CrewAI nhấn mạnh vào việc hợp tác nhóm người-AI.[43] Điều này cho thấy rằng framework "tốt nhất" phụ thuộc nhiều vào nhu cầu kiến trúc cụ thể của ứng dụng. Các mô tả làm nổi bật các trừu tượng hóa cốt lõi và điểm mạnh khác nhau: các tác nhân hội thoại và mẫu của AG2 [1], cấu trúc đồ thị của LangGraph [6], các thành phần của LangChain [88], trọng tâm hợp tác của CrewAI.[43] Sự khác biệt này ngụ ý rằng người dùng nên chọn một framework dựa trên việc thách thức chính của họ là quản lý các cuộc hội thoại phức tạp, xác định các chuyển đổi trạng thái rõ ràng, tích hợp các công cụ LLM khác nhau, hay tạo điều kiện cho các nhóm người-tác nhân.

Hơn nữa, sự hỗ trợ rõ ràng của AG2 cho các công cụ LangChain và CrewAI [42] và khả năng triển khai các tác nhân từ nhiều framework (AG2, LangChain, LangGraph, LlamaIndex) trên các nền tảng như Vertex AI Agent Engine [6] chỉ ra một xu hướng về khả năng tương tác thay vì các hệ sinh thái đơn khối, biệt lập. Người dùng có thể ngày càng kết hợp và phối ghép các thành phần. Việc AG2 xây dựng các cầu nối rõ ràng [42] và các nền tảng triển khai hỗ trợ nhiều framework [6] cho thấy cộng đồng và ngành công nghiệp nhận ra giá trị trong việc tận dụng điểm mạnh từ các công cụ khác nhau. Điều này di chuyển khỏi việc buộc người dùng vào một framework duy nhất và hướng tới một cách tiếp cận mô-đun hơn, nơi các nhà phát triển có thể kết hợp các phần tốt nhất của các hệ sinh thái khác nhau.

Tuy nhiên, cần lưu ý rằng một số so sánh dựa trên thông tin hạn chế hoặc các bình luận chủ quan của người dùng (ví dụ: AG2 dễ học hơn nhưng kém linh hoạt hơn LangGraph trong [86]). Các so sánh khách quan, dựa trên benchmark về hiệu suất, khả năng mở rộng và nỗ lực phát triển thực sự dường như còn thiếu trong các đoạn trích được cung cấp. Đoạn trích [86] trực tiếp trích dẫn ý kiến người dùng về đường cong học tập so với tính linh hoạt. Mặc dù có giá trị về mặt giai thoại, đó không phải là một so sánh nghiêm ngặt. Việc thiếu các benchmark định lượng hoặc phân tích tính năng song song chi tiết trong nghiên cứu được cung cấp có nghĩa là các đánh giá về "tính dễ sử dụng," "tính linh hoạt," hoặc "khả

năng mở rộng" phần lớn vẫn mang tính định tính và có khả năng sai lệch.

7.3. Ma Trận So Sánh Tính Năng AG2 và Các Giải Pháp Thay Thế

Bảng sau cung cấp một ma trận so sánh các tính năng chính giữa AG2 và một số giải pháp thay thế phổ biến, dựa trên thông tin có sẵn:

Tính năng/ Khía cạnh	AG2	LangChain	LangGraph	CrewAI
Mô hình cốt lõi	Hội thoại đa tác nhân [1]	Chuỗi thành phần [88]	Máy trạng thái/Đồ thị [6]	Hợp tác tác nhân [43]
Phong cách điều phối	Mẫu tích hợp (GroupChat, Swarm), Tùy chỉnh [1]	Chain/Agent Executors [88]	Thực thi đồ thị [6]	Ủy thác/Nhiệm vụ tác nhân [62]
Trừu tượng hóa chính	Conversable Agents [1]	Components (LLMs, Prompts, Tools) [88]	Nodes và Edges [6]	Agents, Tasks, Crews [62]
Hỗ trợ thực thi mã	Có (Cục bộ/Docker) [31]	Có (qua tools, rủi ro bảo mật) [62]	Có (trong nodes) [64]	Qua LangChain tools? [62]
Tích hợp công cụ	Có (Tích hợp sẵn, Interop w/ LC/CrewAI) [42]	Thư viện mở rộng [42]	Tích hợp LC tools [6]	Có (Tích hợp sẵn, Interop w/ LC) [42]
Human-in-the-Loop	Có (UserProxyAgent) [1]	Có (Agents, Callbacks) [88]	Có (Human approval nodes) [6]	Có (Tập trung vào hợp tác Người-AI) [43]
Mức độ tùy chỉnh	Cao [22]	Cao (qua LCEL) [88]	Rất cao [86]	Cao [62]
Độ khó học (Chủ quan)	Dễ hơn? (so với LangGraph) [86]	Khó hơn cho non-devs? [88]	Khó hơn? [86]	Tương tự AG2? [62]
Điểm mạnh chính	Tập trung hội thoại, các mẫu [1]	Bộ công cụ rộng, LCEL [88]	Kiểm soát trạng thái rõ ràng, Linh hoạt [6]	Tập trung Người-AI, Xây dựng trên LangChain [62]
Điểm yếu chính	Vấn đề tài liệu, Khả năng mở rộng? [11]	Phức tạp, Chỉ trích "Wrapper" [27]	Phức tạp cho nhiệm vụ đơn giản [86]	Ít tập trung vào tự động hóa thuần túy? [43]

8. Tình Trạng Phát Triển và Lộ Trình Tương Lai

8.1. Phiên Bản Hiện Tại và Tình Trạng Phát Triển

AG2 là một dự án đang được phát triển tích cực:

- **Phiên bản ổn định mới nhất:** v0.9.0, được phát hành vào ngày 25 tháng 4 năm 2025 (ngày này có thể không chính xác nhưng cho thấy một bản phát hành gần đây).[1]
- **Bảo trì:** Được duy trì bởi các tình nguyện viên [1] thuộc tổ chức AG2ai.[5]
- **Hoạt động:** Hoạt động phát triển đang diễn ra, thể hiện qua số liệu GitHub (xem Mục 6.1).
- **Phiên bản:** Trang web tài liệu cung cấp các phiên bản bao gồm 0.9, 0.9dev và các phiên bản cũ hơn.[9]
- **Tính năng thử nghiệm:** Một số tính năng có thể vẫn đang trong giai đoạn thử nghiệm (ví dụ: Ollama client [92]).
- **Tiền thân:** AutoGen v0.4 (tiền thân/liên quan) đã giới thiệu những thay đổi đáng kể như nhắn tin không đồng bộ và tính mô-đun.[25]

Tình trạng này cho thấy AG2 là một dự án năng động với các bản phát hành gần đây, nhưng các vấn đề về tài liệu và các tính năng có khả năng đang thử nghiệm cho thấy nó có thể vẫn đang trong quá trình phát triển và hoàn thiện nhanh chóng.

8.2. Phân Tích Lộ Trình Phát Triển Chính Thức

Một lộ trình phát triển công khai tồn tại, có thể truy cập qua trang web chính (ag2.ai) [10] và có thể thông qua các "roadmap issues" trên GitHub.[84] Các thành phần chính được lên kế hoạch bao gồm:

- **AG2 Studio:** Một môi trường trực quan để thiết kế, kiểm thử, gỡ lỗi và triển khai hệ thống tác nhân, nhằm mục đích tạo mẫu và phát triển sản phẩm khả thi tối thiểu (MVP).[10] AutoGen Studio (tiền thân liên quan) đã cung cấp một giao diện low-code.[63]
- **AG2 Marketplace:** Một nền tảng để chia sẻ, khám phá và có khả năng kiếm tiền từ các tác nhân và mẫu giải pháp.[10]
- **Scaling Tools (Công cụ Mở rộng quy mô):** Các tính năng hỗ trợ triển khai sản xuất, phân tích sử dụng, tối ưu hóa chi phí, hợp tác nhóm và kiểm soát bảo mật cấp doanh nghiệp.[10]

Lộ trình này cho thấy sự tập trung mạnh mẽ vào việc cải thiện tính khả dụng (Studio), thúc đẩy một hệ sinh thái (Marketplace), và cho phép áp dụng trong môi trường sản xuất/doanh nghiệp (Scaling Tools). Điều này phù hợp với tham vọng trở thành một "AgentOS" nền tảng. Sự tập trung của lộ trình vào AG2 Studio (công cụ trực quan), Marketplace (khả năng tái sử dụng), và Scaling Tools (triển khai, phân tích, bảo mật) [10] báo hiệu một sự chuyển dịch chiến lược vượt ra ngoài framework cốt lõi hướng tới việc xây dựng một hệ sinh thái hoàn chỉnh hơn để phát triển *và triển khai* các ứng dụng agentic, đặc biệt là cho những người dùng ít tập trung vào mã lệnh hơn hoặc môi trường doanh nghiệp. Framework cốt lõi cung cấp động cơ. Studio, Marketplace và Scaling Tools giải quyết các nhu cầu xung quanh: thiết kế dễ dàng hơn (Studio), phát triển nhanh hơn thông qua tái sử dụng (Marketplace), và vận hành đáng tin cậy ở quy mô lớn (Scaling Tools). Sự phát triển này là điển hình cho các framework đang trưởng thành nhằm mục đích áp dụng rộng rãi hơn.

8.3. Tình Trạng Projects/Milestones trên GitHub

Kho lưu trữ GitHub có tab "Projects", nhưng hiện tại nó hiển thị "0 Projects".[1] Không có tab

"Milestones" cụ thể nào được đề cập.[1] Chi tiết lộ trình có thể được suy ra từ các "roadmap issues" được đề cập trong hướng dẫn đóng góp [84] hoặc các issues/PRs chung.[1]

Mặc dù một lộ trình chính thức được công bố ở nơi khác [10], các tính năng tiêu chuẩn của GitHub để theo dõi tiến độ (Projects, Milestones) dường như không được sử dụng tích cực để trực quan hóa lộ trình cấp cao tại thời điểm chụp nhanh.[1] Việc theo dõi tiến độ có thể yêu cầu giám sát các issue được gắn thẻ "roadmap".[84]

Việc thực hiện lộ trình tham vọng (Studio, Marketplace, Scaling Tools - [10]) sẽ đòi hỏi nỗ lực phát triển đáng kể. Đối với một dự án do cộng đồng điều khiển dựa vào tình nguyện viên [1], việc đạt được tầm nhìn này có thể phụ thuộc nhiều vào việc thu hút và giữ chân một cơ sở người đóng góp lớn, tích cực hoặc đảm bảo các hình thức hỗ trợ khác. Xây dựng một IDE trực quan (Studio), một thị trường thành phần, và các công cụ mở rộng quy mô cấp doanh nghiệp là những công việc đáng kể. Mặc dù cộng đồng đang hoạt động (Mục 6), quy mô của các hạng mục lộ trình này cho thấy một thách thức tiềm ẩn trong việc cung cấp chúng một cách nhanh chóng và mạnh mẽ chỉ thông qua nỗ lực tình nguyện [1], đặc biệt là với các vấn đề mở hiện có.[1] Thành công có thể phụ thuộc vào hiệu quả của việc xây dựng cộng đồng và mô hình quản trị mở của họ.[5]

Ngoài ra, việc giới thiệu Studio và Marketplace [10] có khả năng dẫn đến các cách xây dựng/định nghĩa tác nhân khác nhau (trực quan so với code-first). Việc đảm bảo tính nhất quán và khả năng tương tác giữa các thành phần được thiết kế trực quan và các tác nhân dựa trên mã sẽ rất quan trọng để tránh phân mảnh hệ sinh thái. Các trình xây dựng trực quan (Studio) thường giới thiệu các trừu tượng hóa riêng của chúng. Một thị trường ngụ ý các thành phần có thể chia sẻ. Nếu các thành phần trực quan hoặc được chia sẻ này không tích hợp liền mạch với framework dựa trên mã cốt lõi, nó có thể tạo ra sự nhầm lẫn và hạn chế tính hữu dụng của các bổ sung mới này. Việc duy trì sự mạch lạc giữa các phương pháp phát triển khác nhau (mã, trực quan, thành phần thị trường) sẽ là một thách thức chính.

9. Kết Luận và Khuyến Nghị

9.1. Tóm Tắt Các Phát Hiện Chính

AG2 (v0.9) nổi lên như một framework mã nguồn mở, do cộng đồng điều khiển, tập trung vào việc xây dựng các ứng dụng AI phức tạp thông qua sự hợp tác của nhiều tác nhân dựa trên hội thoại.

- **Điểm mạnh:**
  - **Mô hình hội thoại đa tác nhân:** Cung cấp các trừu tượng hóa và mẫu (GroupChat, Swarm, Sequential, Nested) mạnh mẽ để điều phối các tương tác phức tạp.[1]
  - **Tùy chỉnh và Linh hoạt:** Lớp ConversableAgent cốt lõi cho phép tùy chỉnh cao hành vi của tác nhân.[22]
  - **Tích hợp Công cụ và Thực thi Mã lệnh:** Khả năng tích hợp các công cụ bên ngoài và thực thi mã lệnh (với tùy chọn Docker an toàn hơn) là yếu tố then chốt cho nhiều ứng dụng thực tế.[1]
  - **Khả năng tương tác:** Hỗ trợ tích hợp công cụ từ các framework phổ biến khác như LangChain và CrewAI.[42]
  - **Human-in-the-Loop:** Thiết kế hỗ trợ tích hợp linh hoạt sự tham gia của con người.[1]
  - **Cộng đồng và Giấy phép:** Có một cộng đồng tích cực, cấu trúc đóng góp rõ ràng và giấy phép Apache 2.0 thân thiện với doanh nghiệp.[1]
- **Điểm yếu và Thách thức:**

    - **Vấn đề Tài liệu:** Việc không thể truy cập nhiều trang tài liệu quan trọng cho phiên bản 0.9 là một rào cản đáng kể, làm tăng đường cong học tập và gây lo ngại về tính ổn định/bảo trì.[11]
    - **Rủi ro Bảo mật Thực thi Mã lệnh:** Mặc dù có Docker, việc thực thi mã do LLM tạo ra vẫn tiềm ẩn rủi ro và đòi hỏi quản lý cẩn thận.[31]
    - **Phụ thuộc vào Chất lượng LLM:** Hiệu quả của một số mẫu (ví dụ: GroupChat) có thể phụ thuộc vào khả năng của LLM được sử dụng.[72]
    - **Khả năng Mở rộng/Linh hoạt:** Có thể kém linh hoạt hơn các cách tiếp cận dựa trên đồ thị (như LangGraph) cho các quy trình công việc rất phức tạp hoặc phi tuyến tính (dựa trên bình luận chủ quan).[86]
    - **Nguồn lực Cộng đồng:** Việc thực hiện lộ trình tham vọng có thể là một thách thức đối với một dự án dựa trên tình nguyện viên.[1]
- **Điểm khác biệt chính:** AG2 tạo sự khác biệt thông qua việc lấy hội thoại làm trung tâm, cung cấp các mẫu điều phối đa tác nhân tích hợp sẵn (đặc biệt là Swarm), và khả năng thực thi mã lệnh mạnh mẽ, đồng thời duy trì cách tiếp cận mã nguồn mở do cộng đồng điều khiển.

9.2. Khuyến Nghị cho Người Dùng Tiềm Năng

Việc lựa chọn AG2 phụ thuộc vào nhu cầu và bối cảnh cụ thể:

- **Phù hợp cho:**
    - Các dự án mà mô hình hợp tác dựa trên hội thoại giữa các tác nhân là tự nhiên và phù hợp (ví dụ: mô phỏng nhóm làm việc, hệ thống hỗ trợ khách hàng nhiều bước).
    - Các nhà phát triển cần các mẫu điều phối tích hợp sẵn như GroupChat hoặc Swarm để quản lý tương tác đa tác nhân mà không cần xây dựng logic điều phối phức tạp từ đầu.
    - Các ứng dụng yêu cầu khả năng thực thi mã lệnh động như một phần của quy trình làm việc của tác nhân (với điều kiện các biện pháp bảo mật, đặc biệt là Docker, được triển khai).
    - Các nhóm muốn tận dụng khả năng tương tác với các công cụ từ LangChain hoặc CrewAI.
    - Những người dùng ưu tiên một dự án mã nguồn mở với quản trị cộng đồng và giấy phép Apache 2.0.
- **Cân nhắc kỹ lưỡng nếu:**
    - **Tài liệu đầy đủ và ổn định là ưu tiên hàng đầu:** Các vấn đề truy cập tài liệu hiện tại cho v0.9 là một rủi ro. Người dùng cần chuẩn bị tinh thần để dựa nhiều hơn vào ví dụ, mã nguồn hoặc tài liệu phiên bản cũ/latest.
    - **Yêu cầu quy trình công việc phi tuyến tính, dựa trên trạng thái phức tạp:** Các framework dựa trên đồ thị như LangGraph có thể cung cấp sự kiểm soát và linh hoạt rõ ràng hơn cho các loại kiến trúc này.[6]
    - **Không thể hoặc không muốn sử dụng Docker:** Việc chạy thực thi mã lệnh cục bộ làm tăng đáng kể rủi ro bảo mật.[31] Nếu Docker không phải là một lựa chọn, cần xem xét kỹ lưỡng các hàm ý bảo mật hoặc hạn chế việc sử dụng tính năng thực thi mã.
    - **Cần sự ổn định tuyệt đối của API:** Tình trạng tài liệu và hoạt động phát triển tích cực có thể cho thấy API vẫn đang được tinh chỉnh. Các dự án yêu cầu sự ổn định API lâu dài có thể cần đánh giá kỹ lưỡng hơn.

9.3. Nhận Xét Chung

AG2 (trước đây là AutoGen) đại diện cho một nỗ lực đáng kể trong việc đơn giản hóa việc xây dựng các hệ thống AI đa tác nhân phức tạp. Cách tiếp cận lấy hội thoại làm trung tâm, cùng với các mẫu

điều phối tích hợp và khả năng thực thi mã lệnh, cung cấp một bộ công cụ mạnh mẽ cho các nhà phát triển và nhà nghiên cứu. Việc chuyển sang mô hình quản trị mở và giấy phép Apache 2.0 báo hiệu một cam kết đối với sự phát triển dựa vào cộng đồng.

Tuy nhiên, dự án đang đối mặt với những thách thức, đặc biệt là về tính hoàn thiện và khả năng truy cập của tài liệu cho phiên bản 0.9. Việc giải quyết các vấn đề này sẽ rất quan trọng để giảm bớt rào cản gia nhập và xây dựng niềm tin của người dùng. Lộ trình đầy tham vọng hướng tới AG2 Studio, Marketplace và Scaling Tools cho thấy một tầm nhìn rõ ràng về việc phát triển thành một hệ sinh thái toàn diện hơn, nhưng việc thực hiện thành công sẽ phụ thuộc vào sức mạnh và sự bền vững của cộng đồng đóng góp.

Nhìn chung, AG2 là một framework đầy hứa hẹn với nền tảng vững chắc và một cộng đồng năng động. Đối với những người sẵn sàng đối mặt với đường cong học tập có thể dốc hơn do các vấn đề về tài liệu và quản lý cẩn thận các rủi ro bảo mật liên quan đến thực thi mã, AG2 cung cấp một cách tiếp cận độc đáo và mạnh mẽ để khai thác sức mạnh của sự hợp tác đa tác nhân trong AI.

Nguồn Trích Dẫn

1. ag2ai/ag2: AG2 (formerly AutoGen): The Open-Source ... - GitHub,https://github.com/ag2ai/ag2
2. Develop an AG2 agent | Generative AI on Vertex AI - Google Cloud,https://cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/develop/ag2
3. Building AI Agents with AutoGen - MLQ.ai,https://blog.mlq.ai/building-ai-agents-autogen/
4. yaronbeen/ag2-1: AG2 (formerly AutoGen): The Open-Source AgentOS. Join us at: https://discord.gg/pAbnFJrkgZ - GitHub,https://github.com/yaronbeen/ag2-1
5. airtai/ag2-mintify: AG2 (formerly AutoGen): The Open-Source AgentOS. Join us at: https://discord.gg/pAbnFJrkgZ - GitHub,https://github.com/airtai/ag2-mintify
6. Develop an agent | Generative AI on Vertex AI - Google Cloud,https://cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/develop/overview
7. AG2 - Key Features,https://docs.ag2.ai/docs/home/quick-start
8. Key Features - AG2,https://docs.ag2.ai/docs/home/home
9. ag2 - AG2,https://docs.ag2.ai/
10. AgentOS,https://ag2.ai/
11. https://docs.ag2.ai/0.9/introduction
12. https://docs.ag2.ai/0.9/user-guide/basic-concepts
13. https://docs.ag2.ai/0.9/user-guide/next-level-concepts
14. https://docs.ag2.ai/0.9/use-cases
15. https://docs.ag2.ai/0.9/quick-start
16. https://docs.ag2.ai/0.9/docs/user-guide/basic-concepts/
17. https://docs.ag2.ai/0.9/docs/user-guide/advanced-concepts/
18. https://docs.ag2.ai/0.9/docs/use-cases/
19. https://docs.ag2.ai/0.9/docs/quick-start/
20. https://docs.ag2.ai/0.9/docs/api-reference/autogen/Agent/
21. https://docs.ag2.ai/0.9/docs/use-cases/notebooks/Notebooks
22. Multi-agent Conversation Framework | AutoGen 0.2,https://microsoft.github.io/autogen/0.2/docs/Use-Cases/agent_chat/
23. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent ...,https://openreview.net/forum?id=BAakY1hNKS
24. AutoGen(AG2) - Introduction,https://docs.together.ai/docs/autogen
25. AutoGen, AG2, and Semantic Kernel: Complete Guide - Towards AI,https://towardsai.net/p/machine-learning/autogen-ag2-and-semantic-kernel-complete-guide
26. Build a financial multi-agent system with AG2 (formerly AutoGen) and Ollama,https://tinztwinshub.com/data-science/build-a-financial-multi-agent-system-with-ag2-and-ollama/
27. Recommendations for AI Agent Frameworks & LLMs for Advanced Agentic Systems : r/AI_Agents - Reddit,https://www.reddit.com/r/AI_Agents/comments/1hzbl20/recommendations_for_ai_agent_frameworks_llms_for/
28. Build a multi-agent RAG system with Granite locally - IBM Developer,https://developer.ibm.com/tutorials/awb-build-agentic-rag-system-granite/
29. LLM Agents: Introduction to Autogen - DEV Community,https://dev.to/admantium/llm-agents-introduction-to-autogen-3emo
30. LLM Agents: Multi-Agent Chats with Autogen - DEV Community,https://dev.to/admantium/llm-agents-multi-agent-chats-with-autogen-2j26
31. Code Execution - AG2,https://docs.ag2.ai/latest/docs/user-guide/advanced-concepts/code-execution/
32. ag2/notebook/agentchat_groupchat_RAG.ipynb at main - GitHub,https://github.com/ag2ai/ag2/blob/main/notebook/agentchat_groupchat_RAG.ipynb

33. Task Solving with Code Generation, Execution and Debugging | AutoGen 0.2,https://microsoft.github.io/autogen/0.2/docs/notebooks/agentchat_auto_feedback_from_code_execution/
34. autogen-example/AutoGen_Example.ipynb at main - GitHub,https://github.com/meyiapir/autogen-example/blob/main/AutoGen_Example.ipynb
35. AutoGen Conversation Patterns - Overview for Beginners - Getting Started with Artificial Intelligence,https://www.gettingstarted.ai/autogen-conversation-patterns-workflows/
36. Conversation Patterns Deep-dive - AG2 docs,https://docs.ag2.ai/docs/user-guide/advanced-concepts/conversation-patterns-deep-dive
37. Getting Started with new Autogen Core API: A Step-by-Step Guide for Developers,https://techcommunity.microsoft.com/blog/azure-ai-services-blog/getting-started-with-new-autogen-core-api-a-step-by-step-guide-for-developers/4290691
38. (Legacy) Implement Swarm-style orchestration with GroupChat - AG2,https://docs.ag2.ai/docs/use-cases/notebooks/notebooks/agentchat_swarm_w_groupchat_legacy
39. Swarm — AutoGen - Microsoft Open Source,https://microsoft.github.io/autogen/dev/user-guide/agentchat-user-guide/swarm.html
40. Conversation Patterns Deep-dive - AG2,https://docs.ag2.ai/docs/user-guide/advanced-concepts/conversation-patterns-deep-dive/
41. ag2/website/docs/topics/swarm.ipynb at main - GitHub,https://github.com/ag2ai/ag2/blob/main/website/docs/topics/swarm.ipynb
42. Cross-Framework LLM Tool Integration with AG2,https://docs.ag2.ai/docs/use-cases/notebooks/notebooks/tools_interoperability
43. Mastering Agentic Conversation Pattern with AG2 - Analytics Vidhya,https://courses.analyticsvidhya.com/courses/mastering-agentic-conversation-pattern-with-ag2
44. Cross-Framework LLM Tool for CaptainAgent - AG2,https://docs.ag2.ai/docs/use-cases/notebooks/notebooks/agentchat_captainagent_crosstool
45. Class AG2Agent (1.87.0) | Generative AI on Vertex AI | Google Cloud,https://cloud.google.com/vertex-ai/generative-ai/docs/reference/python/1.87.0/vertexai.preview.reasoning_engines.AG2Agent
46. Examples | AutoGen 0.2 - Microsoft Open Source,https://microsoft.github.io/autogen/0.2/docs/Examples/
47. Swarm Use Case Example - AG2,https://docs.ag2.ai/docs/user-guide/advanced-concepts/swarm/use-case/
48. Literature Review — AutoGen - Microsoft Open Source,https://microsoft.github.io/autogen/dev//user-guide/agentchat-user-guide/examples/literature-review.html
49. Building Secure AI Agents With Dependency Injection (AG2) - YouTube,https://www.youtube.com/watch?v=HAit1cqNaD0
50. AutoGen, AG2, Agents, Frameworks, Open-source, and Best Practices - YouTube,https://www.youtube.com/watch?v=HI_Z0-IKBjY
51. ag2 - AG2,https://docs.ag2.ai/0.9/
52. Conversation Patterns | AutoGen 0.2 - Microsoft Open Source,https://microsoft.github.io/autogen/0.2/docs/tutorial/conversation-patterns/
53. Code Executors | AutoGen 0.2 - Microsoft Open Source,https://microsoft.github.io/autogen/0.2/docs/tutorial/code-executors/
54. Code execution is now by default inside docker container | AutoGen 0.2,https://microsoft.github.io/autogen/0.2/blog/2024/01/23/Code-execution-in-docker/
55. Building the Operating System for AI Agents - The Data Exchange,https://thedataexchange.media/ag2/

56. Hands-on Guide to Building Multi Agent Chatbots with Autogen - Analytics Vidhya,https://www.analyticsvidhya.com/blog/2024/11/multi-agent-chatbots-with-autogen/
57. AutoGen Conversation Patterns - Complete Overview for Beginners : r/AutoGenAI - Reddit,https://www.reddit.com/r/AutoGenAI/comments/1clrms6/autogen_conversation_patterns_complete_overview/
58. Integrating AI agents to other API's? : r/AI_Agents - Reddit,https://www.reddit.com/r/AI_Agents/comments/1hu92sz/integrating_ai_agents_to_other_apis/
59. Code Execution - AG2,https://docs.ag2.ai/docs/user-guide/advanced-concepts/code-execution
60. Microsoft's AutoGen - A guide to code-executing agents — E2B Blog,https://e2b.dev/blog/microsoft-s-autogen
61. Command Line Code Executors — AutoGen - Microsoft Open Source,https://microsoft.github.io/autogen/stable/user-guide/core-user-guide/components/command-line-code-executors.html
62. CrewAI vs AutoGen for Code Execution AI Agents - E2B,https://e2b.dev/blog/crewai-vs-autogen-for-code-execution-ai-agents
63. AutoGen Studio - Microsoft Open Source,https://microsoft.github.io/autogen/dev/user-guide/autogenstudio-user-guide/index.html
64. LangGraph & Docker: Secure Code Execution for Agents? #698 - GitHub,https://github.com/langchain-ai/langgraph/discussions/698
65. Securing Code in the Era of Agentic AI | Veracode,https://www.veracode.com/blog/securing-code-and-agentic-ai-risk/
66. AI Code Generation: The Risks and Benefits of AI in Software - Legit Security,https://www.legitsecurity.com/blog/ai-code-generation-benefits-and-risks
67. Cybersecurity Risks of AI-Generated Code - CSET,https://cset.georgetown.edu/wp-content/uploads/CSET-Cybersecurity-Risks-of-AI-Generated-Code.pdf
68. 4 AI coding risks and how to address them - Snyk,https://snyk.io/blog/4-ai-coding-risks/
69. Share secrets with a code executor using Docker? · microsoft autogen · Discussion #1663 - GitHub,https://github.com/microsoft/autogen/discussions/1663
70. Mastering Agentic Conversation Pattern with AG2 - Analytics Vidhya,https://www.analyticsvidhya.com/courses/mastering-agentic-conversation-pattern-with-ag2/
71. AutoGen Tutorial | Conversation Patterns Explained | Sequential Chat Agent Workflow,https://www.youtube.com/watch?v=7_LQ3ok35Lc
72. AG2 group chats - only with GPT 4? : r/LLMDevs - Reddit,https://www.reddit.com/r/LLMDevs/comments/1i9w9ip/ag2_group_chats_only_with_gpt_4/
73. Which Local LLMs know best when to speak and when to STFU in group chat agent-to-agent conversations? : r/LocalLLaMA - Reddit,https://www.reddit.com/r/LocalLLaMA/comments/1hy7m1y/which_local_llms_know_best_when_to_speak_and_when/
74. GroupChat - Swarms Docs,https://docs.swarms.world/en/latest/swarms/structs/group_chat/
75. AutoGen Tutorial: Conversation Patterns | Nested Chat Agent Workflow - YouTube,https://www.youtube.com/watch?v=zF_-8OeIPbI
76. The Easiest AutoGen Conversation Patterns Tutorial - YouTube,https://www.youtube.com/watch?v=o-BrxjOlYnc
77. Enhanced Swarms in AG2 - Conditional handoffs, nested chats, and agent state | v0.6,https://www.youtube.com/watch?v=vg2cF9m07Ps
78. AutoGen FULL Tutorial with Python (Step-By-Step) Build AI Agent Teams! - YouTube,https://www.youtube.com/watch?v=V2qZ_lgxTzg
79. microsoft/autogen: A programming framework for agentic AI PyPi: autogen-agentchat Discord:

https://aka.ms/autogen-discord Office Hour: https://aka.ms/autogen-officehour - GitHub,https://github.com/microsoft/autogen
80. ag2ai/build-with-ag2: Sample code and application showcases to get you going with AG2 (formally AutoGen) - GitHub,https://github.com/ag2ai/build-with-ag2
81. Quick Start - AG2,https://docs.ag2.ai/latest/docs/quick-start/
82. Installing AG2,https://docs.ag2.ai/docs/user-guide/basic-concepts/installing-ag2
83. 5 Stock Market API Examples with Python | All built with Autogen - YouTube,https://www.youtube.com/watch?v=RwO12ljrk38
84. Contributing to AG2,https://docs.ag2.ai/docs/contributor-guide/contributing
85. What is the best AI agent framework in Python : r/AI_Agents - Reddit,https://www.reddit.com/r/AI_Agents/comments/1hqdo2z/what_is_the_best_ai_agent_framework_in_python/
86. Langgraph vs CrewAI vs AutoGen vs PydanticAI vs Agno vs OpenAI Swarm : r/LangChain - Reddit,https://www.reddit.com/r/LangChain/comments/1jpk1vn/langgraph_vs_crewai_vs_autogen_vs_pydanticai_vs/
87. AI Agent Roadmap : r/LLMDevs - Reddit,https://www.reddit.com/r/LLMDevs/comments/1jhy53f/ai_agent_roadmap/
88. AI Agent Vs LangChain: In-Depth AI Software Comparison - SmythOS,https://smythos.com/ai-agents/comparison/ai-agent-vs-langchain/
89. Vertex AI Agent Engine overview - Google Cloud,https://cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/overview
90. Vertex AI Agent Builder | Google Cloud,https://cloud.google.com/products/agent-builder
91. AG2 - AI Agent Reviews, Features, Use Cases & Alternatives (2025),https://aiagentsdirectory.com/agent/ag2
92. Model Clients — AutoGen - Microsoft Open Source,https://microsoft.github.io/autogen/dev//user-guide/core-user-guide/components/model-clients.html