BIS - MT · 04/06/2026

Bản Tin AI Hằng Ngày

Cập nhật công nghệ AI mới nhất

> *“It is during our darkest moments that we must focus to see the light.”*
>
> ↪ Chính trong những khoảnh khắc tối tăm nhất, chúng ta phải tập trung để thấy ánh sáng.
>
> — Aristotle

Trong khó khăn cực điểm, tư duy tích cực và kiên trì tìm kiếm giải pháp mới là điều phân biệt người vượt qua và người gục ngã.

TIN TỨC NỔI BẬT

1 Claude Code so sánh với GitHub Copilot 2026: SWE-bench, Giá cả [Đã thử nghiệm]

Claude Code vs GitHub Copilot 2026: SWE-bench, Pricing [Tested]

tech-insider.org Đọc bài viết →

Trong một so sánh gần đây, Claude Code và GitHub Copilot đã được thử nghiệm trong một điểm chuẩn Kỹ thuật Phần mềm (SWE). Kết quả cho thấy cả hai công cụ mã hóa được hỗ trợ bởi AI đều có điểm mạnh và điểm yếu. Trong khi GitHub Copilot vượt trội trong một số lĩnh vực, Claude Code đã thể hiện hiệu suất vượt trội trong những lĩnh vực khác. Bài kiểm tra SWE-bench đã đánh giá khả năng của các công cụ trong việc tạo mã, xác định lỗi và cung cấp đề xuất. Giá cả là một yếu tố quan trọng khác cần xem xét khi chọn giữa hai công cụ. GitHub Copilot cung cấp một kế hoạch miễn phí, cũng như một số cấp độ trả phí, với giá từ $10 đến $100 mỗi tháng. Mặt khác, Claude Code tính phí cố định là $15 mỗi tháng. Hiệu quả về chi phí của mỗi công cụ sẽ phụ thuộc vào nhu cầu và sử dụng cá nhân. So sánh này nhấn mạnh tầm quan trọng của việc xem xét cả hiệu suất và giá cả khi chọn một công cụ mã hóa được hỗ trợ bởi AI. Các nhà phát triển nên cân nhắc lợi ích của từng công cụ và chọn công cụ phù hợp nhất với nhu cầu của họ.

2 Giới thiệu Open Agent Specification (Agent Spec): Một biểu diễn thống nhất cho các agent AI

Introducing the Open Agent Specification (Agent Spec): A Unified Representation for AI Agents

Oracle Blogs Đọc bài viết →

> Oracle đã giới thiệu Open Agent Specification (Agent Spec), một biểu diễn thống nhất cho các tác nhân AI. Agent Spec nhằm cung cấp một khuôn khổ chung cho các nhà phát triển để tạo, tích hợp và tương tác với các tác nhân AI trên nhiều nền tảng và ứng dụng khác nhau. Đặc tả này được thiết kế để thúc đẩy sự phát triển của các hệ thống AI tinh vi và tương tác hơn. Agent Spec định nghĩa một tập hợp các khái niệm và cấu trúc dữ liệu cốt lõi cho phép các tác nhân AI giao tiếp và cộng tác với nhau. Nó bao gồm một định dạng tiêu chuẩn hóa cho siêu dữ liệu tác nhân, hành vi và tương tác, cho phép tích hợp và trao đổi thông tin liền mạch giữa các hệ thống AI khác nhau. Bằng cách áp dụng Agent Spec, các nhà phát triển có thể tạo ra các tác nhân AI có tính mô-đun, tái sử dụng và tương thích với nhiều ứng dụng và nền tảng khác nhau. Điều này có thể dẫn đến việc phát triển nhanh hơn, khả năng mở rộng cải thiện và sự cộng tác tăng cường giữa các hệ thống AI. Agent Spec là một đặc tả mở, giúp nó có thể tiếp cận được với các nhà phát triển và tổ chức trên toàn thế giới, và dự kiến sẽ đóng vai trò quan trọng trong việc định hình tương lai của sự phát triển AI.

3 Tối ưu hóa các quy trình làm việc trên GitHub với AI tạo sinh sử dụng Amazon Bedrock và MCP | Amazon Web Services

Streamline GitHub workflows with generative AI using Amazon Bedrock and MCP | Amazon Web Services

Amazon Web Services (AWS) Đọc bài viết →

> Amazon Web Services (AWS) đã giới thiệu một tích hợp mới cho phép các developer tối ưu hóa quy trình làm việc GitHub của họ bằng cách sử dụng AI tạo sinh. Tích hợp này kết hợp Amazon Bedrock và MCP (Nền tảng Canvas Model), cho phép các developer tận dụng sức mạnh của AI tạo sinh để tự động hóa và tối ưu hóa quy trình làm việc GitHub của họ. Với tích hợp này, các developer có thể sử dụng Amazon Bedrock để tạo và quản lý các model, và sau đó triển khai chúng đến MCP để thực thi. Điều này cho phép các developer tự động hóa các nhiệm vụ như tạo mã, sửa lỗi và phát triển tính năng, làm cho quy trình làm việc GitHub của họ trở nên hiệu quả và năng suất hơn. Tích hợp này cũng cung cấp một trải nghiệm liền mạch cho các developer, cho phép họ làm việc với các kho lưu trữ và quy trình làm việc GitHub hiện có mà không cần thay đổi đáng kể. Bằng cách tận dụng sức mạnh của AI tạo sinh, các developer có thể tập trung vào các nhiệm vụ cấp

cao hơn và tăng tốc quá trình phát triển của họ. Tích hợp này dự kiến sẽ nâng cao trải nghiệm tổng thể của developer và cải thiện hiệu suất của quy trình làm việc GitHub.

4

Uber Giới hạn Sử dụng Công cụ AI như Claude Code để Quản lý Chi phí

Uber Caps Usage of AI Tools Like Claude Code to Manage Costs

Simon Willison Đọc bài viết →

Uber đã triển khai một biện pháp quản lý chi phí để kiểm soát chi tiêu của mình cho các công cụ AI. Trả lời một câu hỏi từ Bloomberg News, một người phát ngôn của Uber cho biết công ty đã đặt giới hạn 1.500 đô la mỗi tháng cho mỗi nhân viên về chi tiêu token cho các công cụ mã hóa AI, chẳng hạn như Claude Code và Cursor. Giới hạn này chỉ áp dụng cho phần mềm mã hóa agentic và không ảnh hưởng đến chi tiêu cho các công cụ khác. Chính sách này nhằm ngăn chặn việc chi tiêu quá mức, điều này đã được nhấn mạnh bởi ngân sách AI của Uber năm 2026 đã bị vượt quá trong bốn tháng. Giới hạn 1.500 đô la mỗi công cụ tương đương với giới hạn hàng năm là 36.000 đô la cho mỗi kỹ sư, tương đương khoảng 11% gói lương trung bình hàng năm cho các kỹ sư phần mềm của Uber tại Hoa Kỳ. Động thái này cho thấy Uber đang gán một giá trị đô la thực tế cho lợi ích mà họ nhận được từ các công cụ AI này.

5

Tích hợp dọc như cơ sở hạ tầng AI: Hệ thống cấy ghép kiến trúc đầy đủ của 21D dạy cho chúng ta về việc xây dựng AI lâm sàng tự động

Vertical integration as AI infrastructure: What 21D's full arch implant system teaches us about building autonomous clinical AI

BD Tech Talks Đọc bài viết →

Một sự phát triển gần đây trong lĩnh vực AI lâm sàng đã dẫn đến việc tạo ra một đường ống tự động cho việc lập kế hoạch phẫu thuật trong việc phục hồi cấy ghép răng toàn miệng. Hệ thống này, được xây dựng bởi 21D, xử lý các giai đoạn tính toán của việc lập kế hoạch phẫu thuật từ đầu đến cuối mà không cần can thiệp của con người. Hệ thống này lấy dữ liệu hình ảnh thô, chạy một chuỗi các bước xử lý được thúc đẩy bởi AI, và tạo ra một hướng dẫn phẫu thuật in 3D sẵn sàng cho phòng phẫu thuật. Khoảng 98% quy trình làm việc được tự động hóa, với vai trò của bác sĩ phẫu thuật bắt đầu từ điểm phẫu thuật. Hệ thống này

giải quyết một tập hợp các vấn đề kỹ thuật cụ thể mà hầu hết các hệ thống AI lâm sàng chưa từng cố gắng. Những vấn đề này bao gồm tích hợp nhiều phương thức dữ liệu, chẳng hạn như quét tomography cone beam (CBCT) và quét trong miệng, vào một mô hình bệnh nhân ba chiều nhất quán. Hệ thống sử dụng phát hiện điểm mốc giải phẫu để đăng ký các nguồn dữ liệu này vào một không gian tọa độ thống nhất, cho phép tự động hóa hạ lưu. Cách tiếp cận này cho phép tạo ra một bản sao kỹ thuật số của bệnh nhân, một mô hình ba chiều tổng hợp kết hợp hình học xương và giải phẫu nội bộ với hình học bề mặt và chi tiết khớp cắn.

6

Cách Xây dựng Một Agent Harness Tùy chỉnh

How to Build a Custom Agent Harness

LangChain Blog Đọc bài viết →

Hàm create_agent của LangChain là một nguyên thủy để xây dựng một khung tùy chỉnh xung quanh một model, cho phép tùy chỉnh chi tiết. Không giống như các khung đã được lắp ráp sẵn như Deep Agents và Claude Agent SDK, create_agent sử dụng cách tiếp cận tối giản,□□ middleware như một nguyên thủy để tùy chỉnh. Middleware có thể được sử dụng để thêm các khả năng như logic quyết định, logic kinh doanh và quản lý công cụ, cũng như trạng thái và xử lý luồng tùy chỉnh. Cách tiếp cận này cho phép các developer tự do kết hợp middleware để phù hợp với nhu cầu cụ thể của họ. LangChain cung cấp middleware đã được xây dựng sẵn cho các mẫu thông dụng, và middleware tùy chỉnh có thể được tạo cho các trường hợp sử dụng độc đáo. Mục tiêu của một khung là cung cấp cho model đúng ngữ cảnh vào đúng thời điểm cho một nhiệm vụ nhất định, và bảng cung cấp ánh xạ các khả năng thông dụng đến middleware hỗ trợ chúng. Bằng cách sử dụng create_agent và middleware, các developer có thể xây dựng các agent tùy chỉnh phù hợp với nhu cầu của nhiệm vụ cụ thể của họ.

7

MCP trên Chế độ Code

MCP on Code Mode

Changelog Đọc bài viết →

Trong một tập gần đây, Matt Carey từ Cloudflare đã thảo luận về Code Mode và mối quan hệ của nó với MCP (Model-Code-Protocol). Carey

đã tiết lộ rằng nhiều người đã hiểu lầm về MCP, và rằng chế độ Code Mode phía máy chủ cho phép một máy chủ MCP duy nhất□ lộ hơn 2.500 điểm cuối API của Cloudflare bằng cách sử dụng khoảng 1.000 token ngữ cảnh. Trình tải Worker động tại Cloudflare chạy an toàn mã được viết bởi model trong một V8 isolate. Carey cũng đã chia sẻ quy trình làm việc cá nhân của mình với Claude, một model, và thảo luận về vai trò của bộ nhớ trong tương lai của các agent. Ngoài ra, ông cũng đã đề cập đến việc sử dụng một công cụ bao git gọi là Zaggy để ngăn chặn force-pushes đến các kho lưu trữ của mình.

8 Các tiền tố phụ thuộc là rủi ro trong chuỗi cung ứng: hãy sửa chúng

Dependency prefixes are a supply chain risk: let's fix them

Sourcegraph Blog Đọc bài viết →

Một mối quan ngại gần đây đã được đề cập liên quan đến việc sử dụng tiền tố phụ thuộc trong phát triển phần mềm. Những tiền tố này, được biểu thị bằng các ký hiệu như ^ và ~, được thiết kế để làm cho việc cập nhật dễ dàng hơn bằng cách chỉ định phạm vi phiên bản cho các phụ thuộc. Tuy nhiên, cách tiếp cận này có một hạn chế đáng kể. Phạm vi phiên bản được tạo bởi những tiền tố này có thể vô tình mở rộng con đường mà một gói bị xâm phạm có thể đi vào sản xuất, gây ra rủi ro cho toàn bộ chuỗi cung ứng. Vấn đề này phát sinh vì phạm vi phiên bản có thể bao gồm không chỉ phiên bản được chỉ định mà còn cả các bản cập nhật tiếp theo, có khả năng cho phép một gói bị xâm phạm được giới thiệu vào môi trường sản xuất. Do đó, các nhà phát triển và tổ chức đang được khuyến cáo nên□□ đánh giá việc sử dụng tiền tố phụ thuộc và xem xét các cách tiếp cận thay thế để quản lý phụ thuộc, nhằm giảm thiểu rủi ro này và đảm bảo an ninh cho chuỗi cung ứng phần mềm của họ.

TIPS & TRICKS CHO DEV

Tối ưu hóa RAG

Vấn đề: Tốc độ xử lý chậm khi sử dụng RAG.

Cách làm: Sử dụng kỹ thuật caching và tối ưu hóa mô hình để tăng tốc độ. Ví dụ, sử dụng lệnh `torch.cuda.empty_cache()` để giải phóng bộ nhớ.

Đánh giá: Hiệu quả cao khi xử lý lượng lớn dữ liệu.

Tìm kiếm ngữ nghĩa

Vấn đề: Tìm kiếm thông tin không chính xác do không hiểu ngữ nghĩa.

Cách làm: Sử dụng semantic search với embeddings như Sentence-BERT. Ví dụ, sử dụng câu prompt "What is the meaning of life?" với mô hình `sentence-transformers`.

Đánh giá: Hiệu quả cao khi tìm kiếm thông tin liên quan đến ngữ nghĩa.

Kết hợp Retrieval-Augmented Generation

Vấn đề: Không thể tạo nội dung mới dựa trên thông tin đã có.

Cách làm: Kết hợp RAG với mô hình sinh nội dung như T5. Ví dụ, sử dụng lệnh `t5-generate --prompt "Write a story about..."` với tùy chọn `--retrieval-augmented`.

Đánh giá: Hiệu quả cao khi tạo nội dung mới dựa trên thông tin đã có.

BÀI HỌC AI HÔM NAY CHO DEV

1. **Tối ưu chi phí & hiệu năng LLM**
2. Dev cần biết về tối ưu chi phí và hiệu năng LLM để giảm thiểu chi phí vận hành và cải thiện hiệu suất của ứng dụng. Điều này giúp đảm bảo rằng mô hình AI hoạt động hiệu quả và tiết kiệm tài nguyên. Việc tối ưu hóa cũng giúp giảm thiểu tác động môi trường của các mô hình AI lớn.
3. Ví dụ, sử dụng kỹ thuật fine-tuning và LoRA (Low-Rank Adaptation) có thể giúp giảm kích thước mô hình và tăng tốc độ xử lý.
4. Tip: Sử dụng các công cụ như Hugging Face Transformers để tối ưu hóa và tinh chỉnh mô hình LLM, giúp giảm chi phí và cải thiện hiệu suất ứng dụng.