

KÊNH ĐI LÀM IT

Kiến thức IT và nghề nghiệp

BÁO CÁO TÌM HIỂU · TÀI LIỆU ĐI KÈM VIDEO

JEV — MÔ HÌNH AI KHÔNG VIẾT CHỮ

*Giao diện tính toán, hiệu chuẩn xác suất và các đánh giá độc lập
của một mô hình quyết định kiểu System One*

Video YouTube: Jev — mô hình AI không viết chữ (16:18)

Thực hiện: Kênh Đi Làm IT

Thành phố Hồ Chí Minh, tháng 10 năm 2026

MỤC LỤC

1. Giới thiệu.....	3
2. Bối cảnh: mô hình ngôn ngữ sinh chữ như thế nào.....	4
2.1. Giải mã tự hồi quy	4
2.2. Đầu ra có cấu trúc chỉ là văn bản được phân tích	4
2.3. Xác suất token không phải độ tin cậy của quyết định.....	5
3. Jev: giao diện và cách tính.....	5
3.1. Ba kiểu câu hỏi.....	5
3.2. Công thức độ tin cậy	6
3.3. Nhiều câu hỏi song song	6
3.4. So sánh với LLM và bộ phân loại cổ điển	7
4. Kiến trúc và huấn luyện: biết gì, đoán gì.....	7
4.1. Những gì đã được công bố.....	7
4.2. Các giả thuyết	7
4.3. RLCD so với RLHF và RLVR.....	8
5. Hiệu chuẩn xác suất.....	8
5.1. Định nghĩa và thước đo.....	8
5.2. Kết quả đánh giá độc lập.....	9
5.3. Ngưỡng quan trọng không kém mô hình	10
5.4. Dự đoán có chọn lọc	11
6. Giới hạn	11
6.1. Ngôn ngữ và nhãn mìn.....	11
6.2. Nghịch lý MMLU và nguy cơ nhiễm dữ liệu	11
6.3. Bị lái bởi đầu vào	12
7. Hệ sinh thái và đối thủ.....	13
8. Khuyến nghị sử dụng.....	14
9. Kết luận	14
Tài liệu tham khảo	15

TÓM TẮT

Jev là mô hình thương mại của TypeSafe AI, ra mắt ngày 15/9/2026, được giới thiệu như một mô hình "System One": thay vì sinh văn bản như các mô hình ngôn ngữ lớn (LLM), Jev nhận một trạng thái đầu vào cùng các câu hỏi có kiểu, và trả về một phân phối xác suất trên tập lựa chọn do người gọi cung cấp. Báo cáo này phân tích giao diện tính toán của Jev, so sánh với LLM dạng decoder và bộ phân loại cổ điển, tách bạch những gì đã được công bố với những gì chỉ là giả thuyết về kiến trúc, và tổng hợp các đánh giá độc lập. Trọng tâm là **hiệu chuẩn xác suất** (calibration): đánh giá độc lập trên 37 bộ dữ liệu cho thấy Jev hiệu chuẩn tốt với câu hỏi lựa chọn (ECE gộp 0,028), nhưng dự đoán "có" quá nhiều ở bài toán đa nhãn, phụ thuộc mạnh vào ngưỡng quyết định, suy giảm ở ngôn ngữ ít tài nguyên và dễ bị lái bởi nội dung chèn vào đầu vào. Báo cáo kết thúc bằng các khuyến nghị sử dụng cho kỹ sư phần mềm.

Từ khoá: Jev, mô hình quyết định, hiệu chuẩn xác suất, ECE, selective prediction, LLM.

1. GIỚI THIỆU

Phần lớn ứng dụng AI hiện nay dựa vào mô hình ngôn ngữ lớn sinh văn bản. Tuy nhiên, rất nhiều bước trong một hệ thống thực tế không cần văn bản mà cần một **quyết định**: email nên chuyển cho phòng nào, lệnh gọi công cụ có an toàn không, câu trả lời có bịa thông tin không. Dùng LLM cho các quyết định nhỏ này tốn thời gian, chi phí, và kết quả phải được phân tích cú pháp từ văn bản.

Jev [1], [3] đặt cược vào một giao diện khác: mô hình chỉ trả về phân phối xác suất trên các lựa chọn hữu hạn, trong khoảng một phần ba giây, với giá 0,042 USD cho một triệu token đầu vào và đầu ra miễn phí [3], [29]. Báo cáo này trả lời bốn câu hỏi: (1) Jev tính toán cái gì và khác LLM ở đâu; (2) những gì đã biết và chưa biết về kiến trúc, huấn luyện; (3) xác suất của Jev có đáng tin không; (4) khi nào nên dùng. Với mỗi con số, báo cáo nêu rõ **ai đo, đo trên cái gì và so với ai**, đồng thời phân biệt số liệu của nhà sản xuất với đánh giá độc lập.

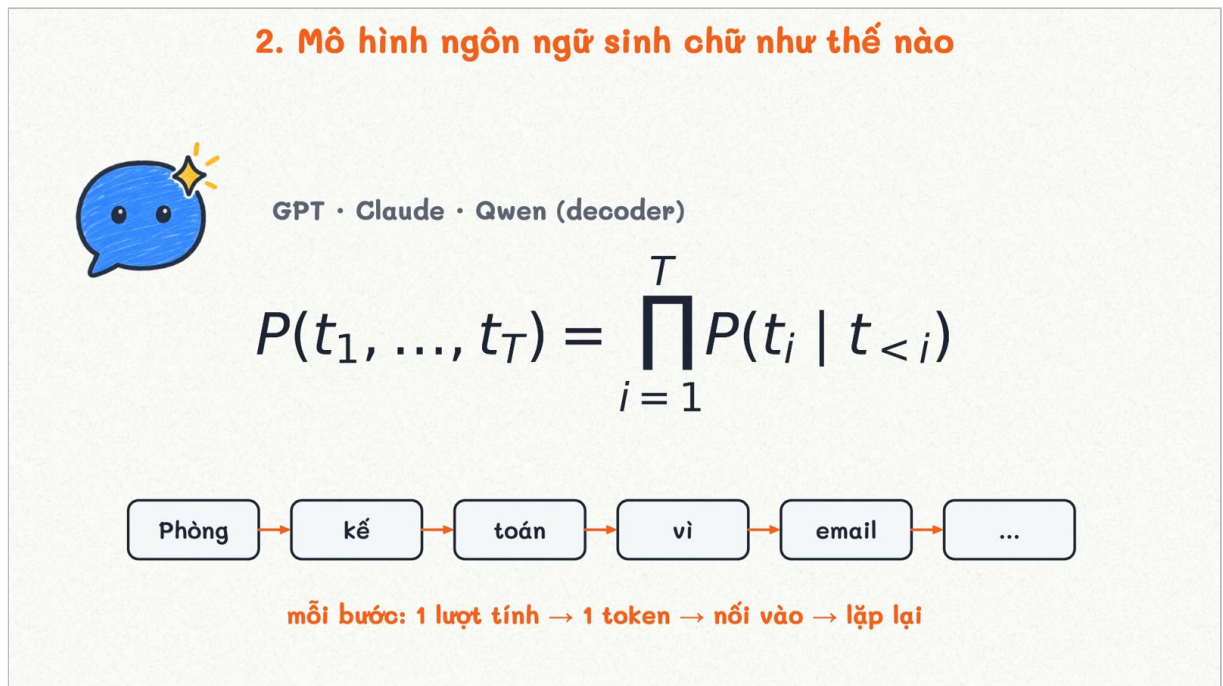
2. BỐI CẢNH: MÔ HÌNH NGÔN NGỮ SINH CHỮ NHƯ THẾ NÀO

2.1. Giải mã tự hồi quy

Một LLM dạng decoder (GPT, Claude, Qwen) học phân phối có điều kiện của token tiếp theo khi biết các token trước đó. Xác suất của cả chuỗi đầu ra là tích các xác suất có điều kiện:

$$P(t_1, \dots, t_T) = \prod_{i=1}^T P(t_i | t_{<i})$$

Khi sinh câu trả lời, mô hình chạy một lượt tính toán cho mỗi token: chọn một token, nối vào chuỗi, rồi lặp lại (Hình 1). Hệ quả là độ trễ gần như tăng tuyến tính theo độ dài đầu ra; với mô hình suy luận, đầu ra còn gồm hàng nghìn token "suy nghĩ" trước khi có câu trả lời.



Hình 1. Quá trình sinh từng token của LLM dạng decoder (khung hình tại 2:38 trong video)

2.2. Đầu ra có cấu trúc chỉ là văn bản được phân tích

Khi yêu cầu LLM trả về JSON, mô hình vẫn đang viết chữ; tính đúng định dạng đến từ prompt, ràng buộc khi giải mã và bộ phân tích cú pháp, chứ không phải từ không gian đầu ra. Vì vậy vẫn có lỗi sai kiểu, bịa trường hoặc từ chối bằng văn bản. Số liệu do TypeSafe tổng hợp từ OpenRouter (hãng tự ghi có thiên lệch) cho tỉ lệ lỗi đầu ra có cấu trúc từ 0,58% đến 45,5% tùy mô hình [3].

2.3. Xác suất token không phải độ tin cậy của quyết định

Xác suất mô hình gán cho chữ "kế toán" không đảm bảo rằng, trong những lần tự tin như vậy, mô hình đúng với tỉ lệ tương ứng. Các nghiên cứu được tổng quan trong [5] cho thấy huấn luyện theo phản hồi người dùng (RLHF) thường làm xác suất của mô hình kém trung thực hơn.

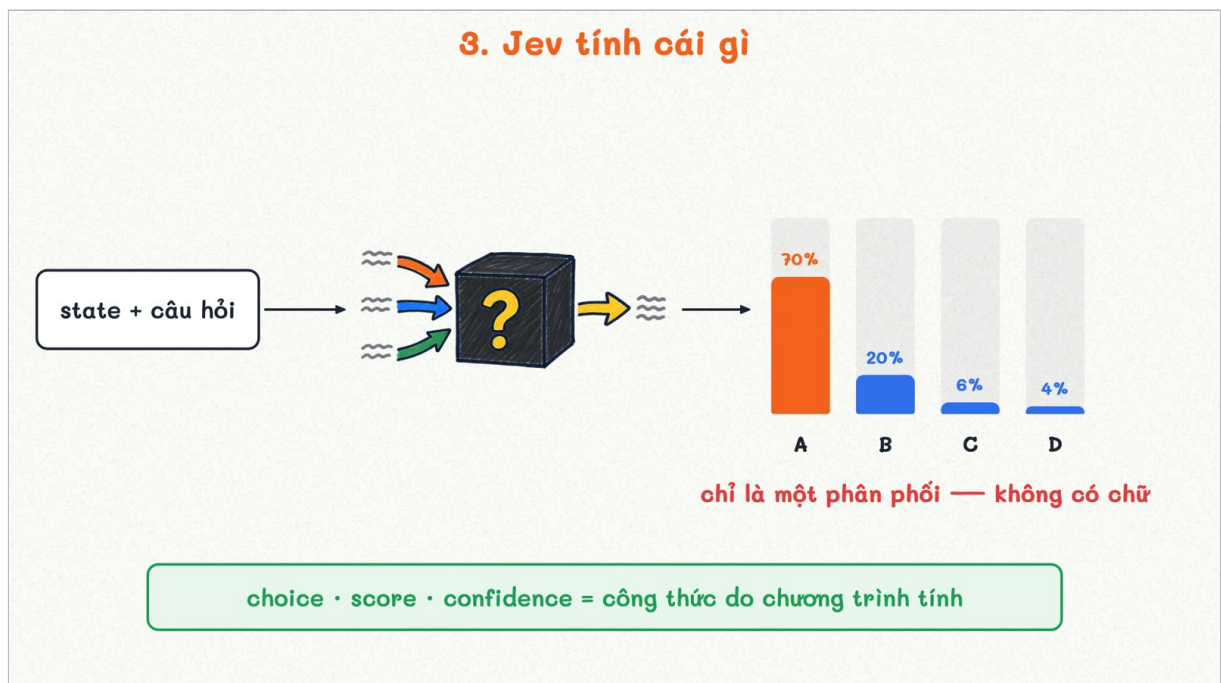
3. JEV: GIAO DIỆN VÀ CÁCH TÍNH

3.1. Ba kiểu câu hỏi

Jev có một điểm gọi API duy nhất (POST /v1/systemone). Đầu vào gồm một `state` (chuỗi, JSON hoặc mảng) và một hoặc nhiều câu hỏi có kiểu [26]:

- **Choice:** chọn một trong tối đa 255 phương án do người gọi đưa ra;
- **Score:** đặt đầu vào lên một thang từ 2 đến 10 mức;
- **Noul:** xác suất trong khoảng [0, 1] để một phát biểu là đúng.

Đầu ra duy nhất của mô hình là một phân phối xác suất rời rạc cho mỗi câu hỏi. Các giá trị API trả về (`choice`, `score`, `confidence`) là công thức cố định do chương trình tính từ phân phối đó, không phải do mô hình viết ra [1], [27] (Hình 2).



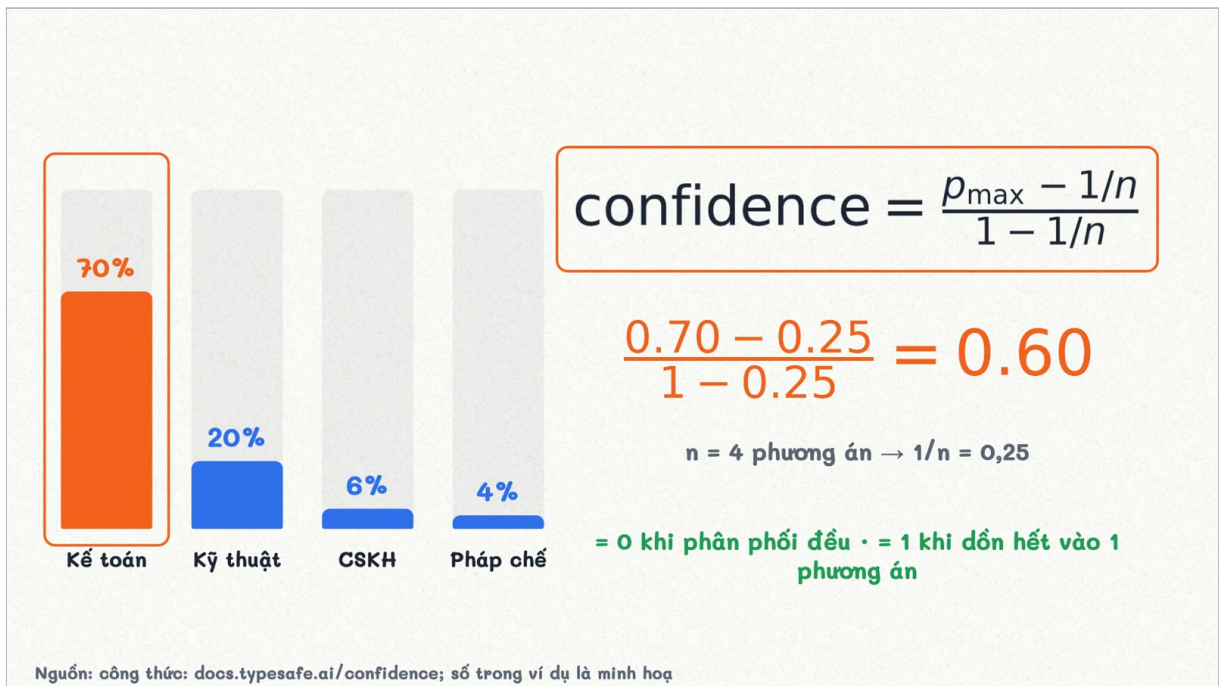
Hình 2. Luồng tính của Jev: `state` và câu hỏi đi vào, phân phối xác suất đi ra (khung hình tại 4:18 trong video)

3.2. Công thức độ tin cậy

Với câu hỏi Choice có n phương án và xác suất lớn nhất $p_{\max}$, độ tin cậy được tài liệu của TypeSafe định nghĩa là [27]:

$$\text{confidence} = \frac{p_{\max} - 1/n}{1 - 1/n}$$

Giá trị bằng 0 khi phân phối đều (mô hình hoàn toàn không biết) và bằng 1 khi toàn bộ xác suất dồn vào một phương án. Ví dụ với bốn phương án và phân phối (0,70; 0,20; 0,06; 0,04), phương án được chọn là "Kế toán" và độ tin cậy là $(0,70 - 0,25)/(1 - 0,25) = 0,60$ (Hình 3).



Hình 3. Ví dụ tính độ tin cậy của câu hỏi Choice (khung hình tại 4:42 trong video)

Với Score, điểm là giá trị kỳ vọng $\text{score} = \sum i \cdot p_i$; ví dụ thang ba mức với phân phối (0,05; 0,80; 0,15) cho điểm 1,10. Với Noul, API **không** trả độ tin cậy; tài liệu gợi ý người dùng tự tính $|2p - 1|$ [27]. Cần lưu ý một mâu thuẫn: hình minh họa trong bài blog của LangChain [22] ghi độ tin cậy 0,596 cho phân phối (0,84; 0,159; 0,001), trong khi công thức tài liệu cho 0,76; con số 0,596 lại khớp xấp xỉ với "1 - entropy chuẩn hoá". Bài JevAdvBench (truy cập tài liệu ngày 24/9) còn ghi công thức chưa được công bố [17]. Người dùng nên tự kiểm tra lại.

3.3. Nhiều câu hỏi song song

Các câu hỏi trên cùng một state được trả lời song song và độc lập; thêm câu hỏi gần như không làm tăng độ trễ, chỉ tốn thêm token đầu vào. Trong đánh giá độc lập, một request có 28 câu Noul tốn trung bình 765 token so với 559 token cho một câu

phân loại; độ trễ trung bình đo phía người gọi (gồm mạng) là 0,36 giây [5]. Đổi lại, các câu hỏi không nhìn thấy nhau: nếu câu B phụ thuộc đáp án câu A, phải gọi hai lần.

3.4. So sánh với LLM và bộ phân loại cổ điển

Bảng 1 tóm tắt khác biệt. Bộ phân loại kiểu BERT cũng trả xác suất trong một lượt tính, nhưng nhận bị cố định lúc huấn luyện; Jev nhận nhãn và tiêu chí ngay lúc gọi (zero-shot). Vì không thể trả giá trị ngoài các phương án, Jev không bao giờ sai kiểu, nhưng vẫn có thể chọn sai — đó là lỗi phân loại, không phải lỗi định dạng.

Bảng 1. So sánh giao diện tính toán

	LLM (decoder)	BERT + lớp phân loại	Jev
Đầu ra	Chuỗi token	Phân phối trên nhãn đã huấn luyện	Phân phối trên lựa chọn đưa lúc gọi
Thêm nhãn mới	Đổi prompt	Huấn luyện lại	Gửi kèm trong request
Lỗi sai kiểu	Có thể xảy ra	Không	Không (theo cấu tạo)
Giải thích bằng lời	Có	Không	Không
Chạy cục bộ	Tuỳ mô hình	Có	Không — chỉ qua API

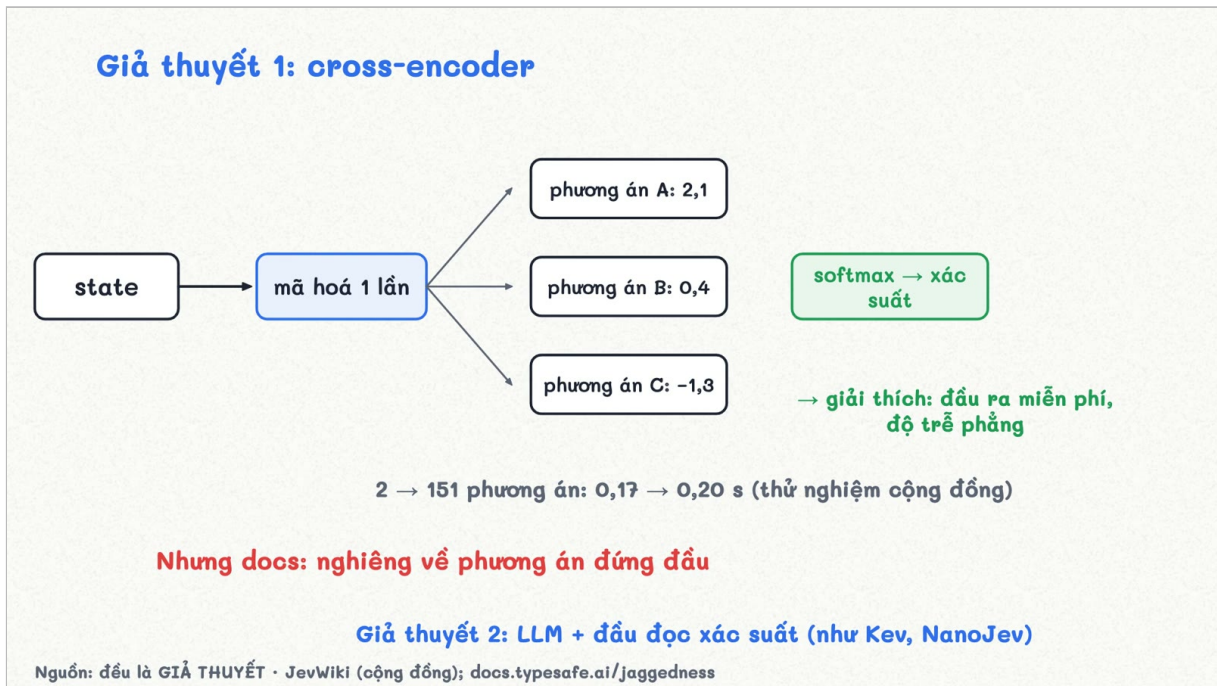
4. KIẾN TRÚC VÀ HUẤN LUYỆN: BIẾT GÌ, ĐOÁN GÌ

4.1. Những gì đã được công bố

TypeSafe chỉ mô tả hệ thống gồm ba thành phần: một kiến trúc mô hình mới, một bộ lấy mẫu song song (parallel sampler) và phương pháp huấn luyện RLCD [3]. Số tham số, trọng số, dữ liệu huấn luyện và bài báo kỹ thuật chưa được công bố [1], [20]. Mô tả "Jev chạy trong đúng một lượt tính toán" xuất hiện ở nhiều bài viết nhưng là suy luận của người ngoài.

4.2. Các giả thuyết

Giả thuyết 1 — cross-encoder: state được mã hoá một lần, mỗi phương án được chấm điểm như một giả thuyết rồi chuẩn hoá bằng softmax (Hình 4). Giả thuyết này giải thích vì sao đầu ra miễn phí và vì sao, theo một thử nghiệm cộng đồng, độ trễ gần như không đổi khi số phương án tăng từ 2 lên 151 [10]. Tuy nhiên tài liệu của chính hãng thừa nhận mô hình nghiêng về phương án đứng đầu [28], tức các phương án không hoàn toàn độc lập với vị trí.



Hình 4. Giải thuyết cross-encoder: mã hoá state một lần, chấm từng phương án (khung hình tại 7:38 trong video)

Giải thuyết 2 — LLM kèm đầu đọc xác suất: một mô hình ngôn ngữ được gắn thêm lớp đọc phân phối ở một vị trí đặc biệt, giống cách các bản tái tạo mã nguồn mở như Kev, NanoJev đang làm [12], [13].

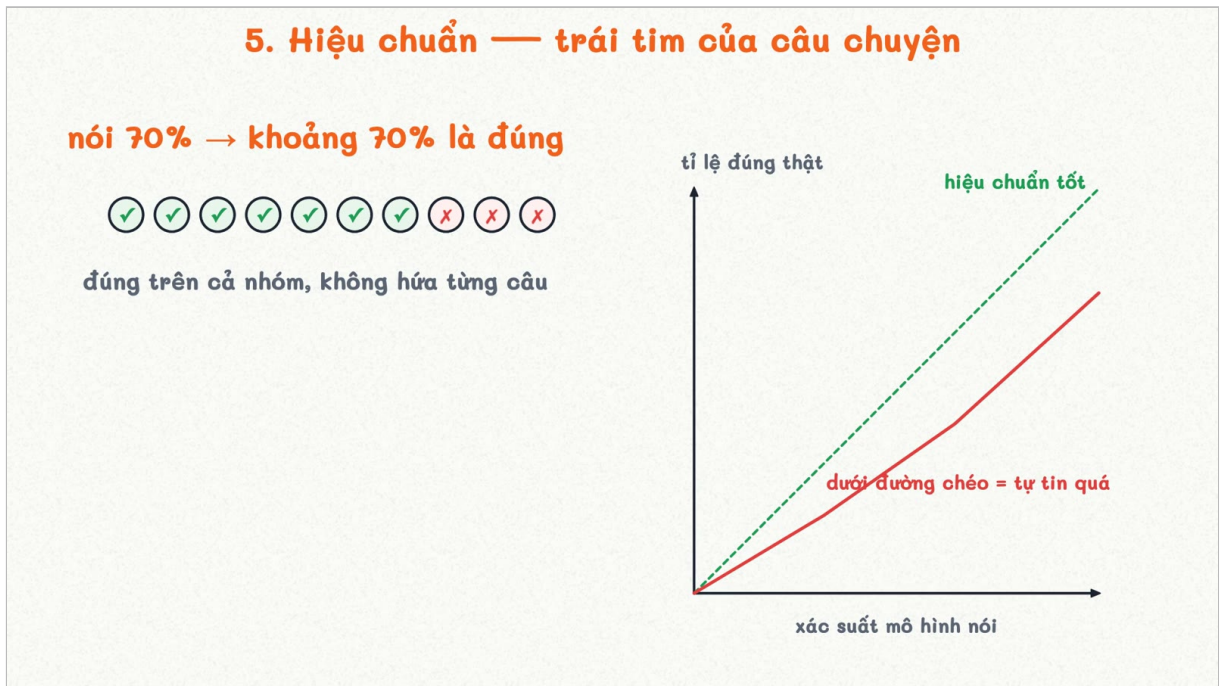
4.3. RLCD so với RLHF và RLVR

RLHF thường câu trả lời người chấm ưa thích; RLVR thường đầu ra kiểm chứng được bằng chương trình. RLCD, theo TypeSafe, thường những quyết định có xác suất trung thực: nói 90% thì đúng khoảng 90% [3], [30]. Hàm mất mát và dữ liệu không được công bố. Một thí nghiệm cộng đồng (RSI-Jev) thấy phần thưởng học tăng cường làm hiệu chuẩn xấu đi, trong khi một đầu hiệu chỉnh đơn giản phía sau khôi phục gần hết [20] — gợi ý rằng hiệu chuẩn tốt chưa chắc đến từ đúng chỗ hằng mô tả.

5. HIỆU CHUẨN XÁC SUẤT

5.1. Định nghĩa và thước đo

Một mô hình được hiệu chuẩn tốt khi, trong tất cả các dự đoán có độ tin cậy 70%, khoảng 70% là đúng. Hiệu chuẩn là tính chất của cả nhóm dự đoán, không đảm bảo từng câu trả lời đúng [30]. Biểu đồ độ tin cậy (reliability diagram) đặt độ tin cậy lên trục hoành và tỉ lệ đúng lên trục tung; nằm dưới đường chéo nghĩa là tự tin quá mức (Hình 5).



Hình 5. Biểu đồ độ tin cậy: đường chéo là hiệu chuẩn hoàn hảo (khung hình tại 9:08 trong video)

Sai số hiệu chuẩn kỳ vọng (ECE) chia dự đoán thành các khoảng B_b theo độ tin cậy và lấy trung bình có trọng số của chênh lệch giữa tỉ lệ đúng và độ tin cậy; Brier score là trung bình bình phương khoảng cách giữa xác suất và kết quả thật ($y \in \{0, 1\}$):

$$ECE = \sum_b \frac{|B_b|}{N} | \text{acc}(B_b) - \text{conf}(B_b) |$$

$$\text{Brier} = \frac{1}{N} \sum_i (p_i - y_i)^2$$

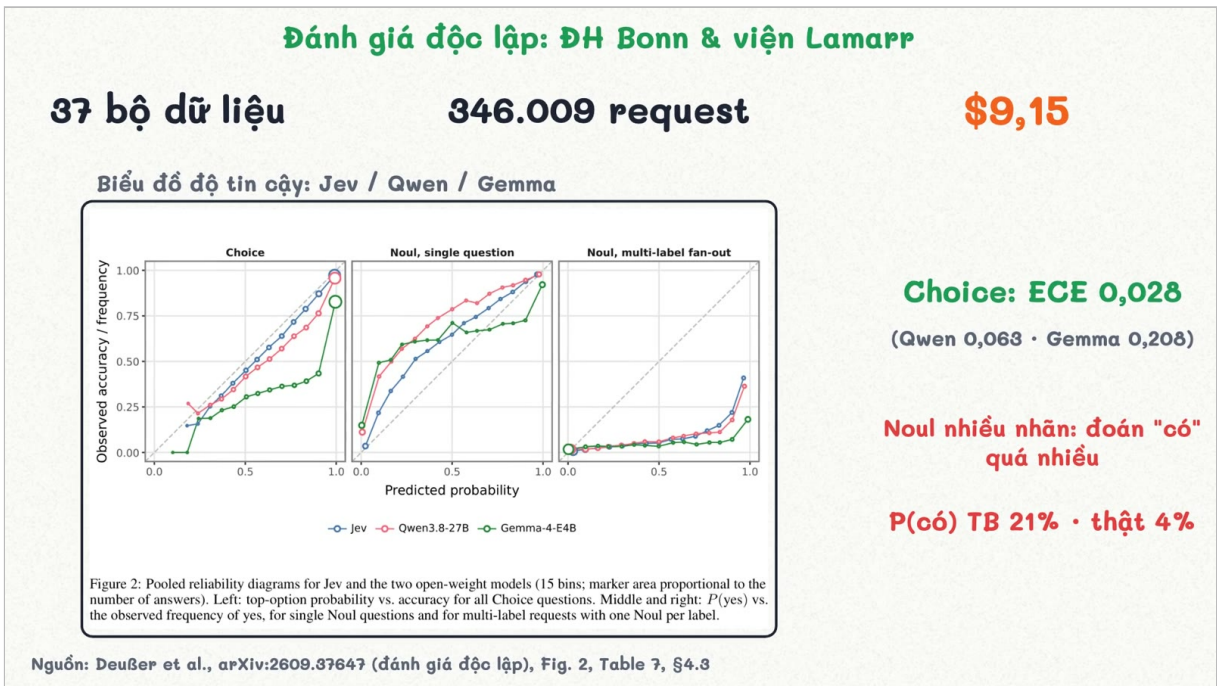
5.2. Kết quả đánh giá độc lập

Đánh giá độc lập lớn nhất tới nay do Deußer, Sparrenberg và Sifa (Đại học Bonn và viện Lamarr) thực hiện trên 37 bộ dữ liệu, 346.009 request, tổng chi phí 9,15 USD, so với Qwen3.8-27B và Gemma-4-E4B chạy trên cùng request nhưng tắt chế độ suy luận [5]. Một số kết quả chính:

Bảng 2. Kết quả hiệu chuẩn và độ chính xác chọn lọc (nguồn: [5])

Chỉ số	Jev	Qwen3.8-27B	Gemma-4-E4B
ECE gộp, câu hỏi Choice	0,028	0,063	0,208
ECE, Noul đa nhãn	0,168	0,119	0,161
Số bộ thắng (trên 37)	—	Jev hơn ở 27	Jev hơn ở 37
Banking77, toàn bộ câu	79,7%	77,4%	67,0%
Banking77, 50% câu tự tin nhất	96,3%	94,5%	—

Với Noul đa nhãn, khi mỗi nhãn được hỏi riêng, Jev dự đoán "có" quá nhiều: xác suất "có" trung bình 20,9% trong khi tỉ lệ thật chỉ 4,1% [5] (Hình 6).



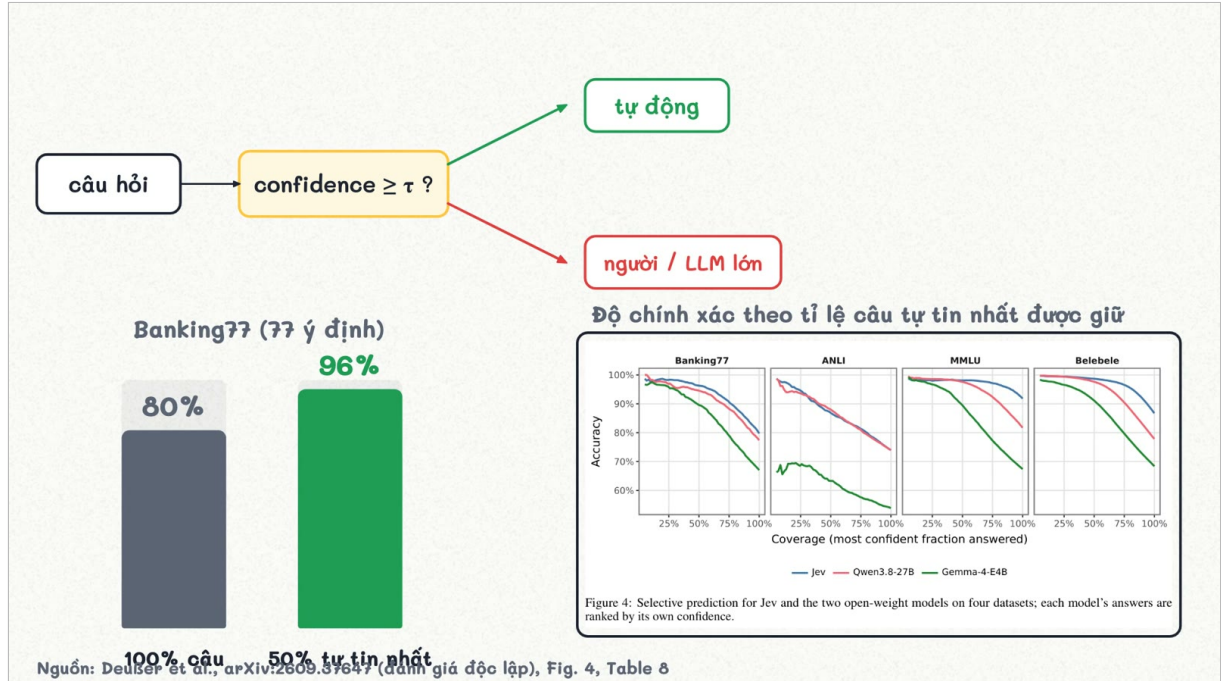
Hình 6. Kết quả hiệu chuẩn trong đánh giá độc lập của Đại học Bonn (khung hình tại 10:13 trong video)

5.3. Ngưỡng quan trọng không kém mô hình

Ngưỡng mặc định 0,5 có thể làm kết quả rất kém. Trên UNFAIR-ToS (điều khoản bất công trong hợp đồng), F1 của Jev tăng từ 0,499 lên 0,748 khi ngưỡng được chọn lại trên 1.000 mẫu có nhãn [5]. Arize ghi nhận điều tương tự khi dùng Jev phát hiện ảo giác: ở ngưỡng 0,5, Jev đúng 76% so với 83% của Claude Opus 5; sau khi tinh chỉnh ngưỡng, cả hai cùng đạt 87%, và Jev rẻ hơn khoảng 300 lần [6] (số của đối tác, phương pháp được công bố).

5.4. Dự đoán có chọn lọc

Hiệu chuẩn tốt cho phép chỉ tự động hoá những câu mô hình tự tin, phần còn lại chuyển cho người hoặc mô hình lớn hơn (Hình 7). Trên Banking77, giữ 50% câu tự tin nhất nâng độ chính xác từ 79,7% lên 96,3% [5].



Hình 7. Dự đoán có chọn lọc: tự động khi tự tin, chuyển người khi không (khung hình tại 11:32 trong video)

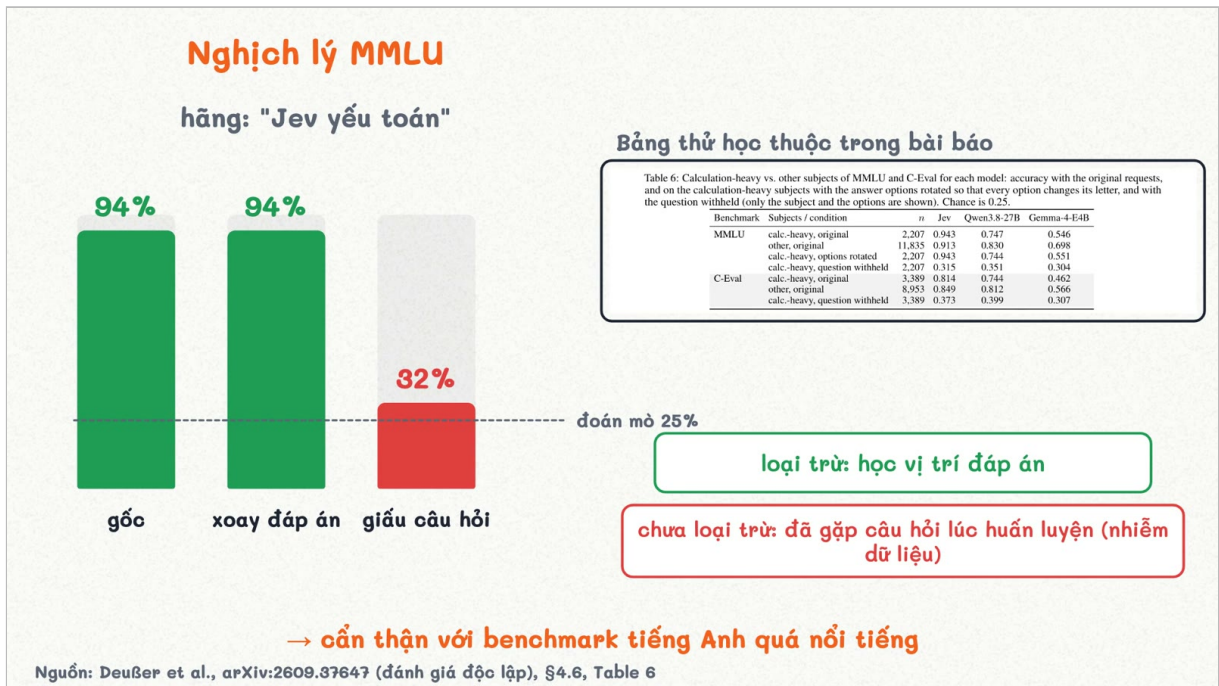
6. GIỚI HẠN

6.1. Ngôn ngữ và nhãn mìn

Trên bộ đọc hiểu Belebele, Jev đạt 97,2% với tiếng Anh, trung vị 91,1% trên 122 biến thể ngôn ngữ, nhưng chỉ 37,4% ở ngôn ngữ thấp nhất [5]; hăng nói rõ tiếng Anh là ngôn ngữ chính. Với nhãn mìn, các thử nghiệm cộng đồng trên Banking77 cho thấy mô hình BERT huấn luyện riêng đạt khoảng 94% so với khoảng 80% của Jev; trên các tác vụ pháp lý, Jev cũng kém mô hình huấn luyện riêng vài điểm [10].

6.2. Nghịch lý MMLU và nguy cơ nhiễm dữ liệu

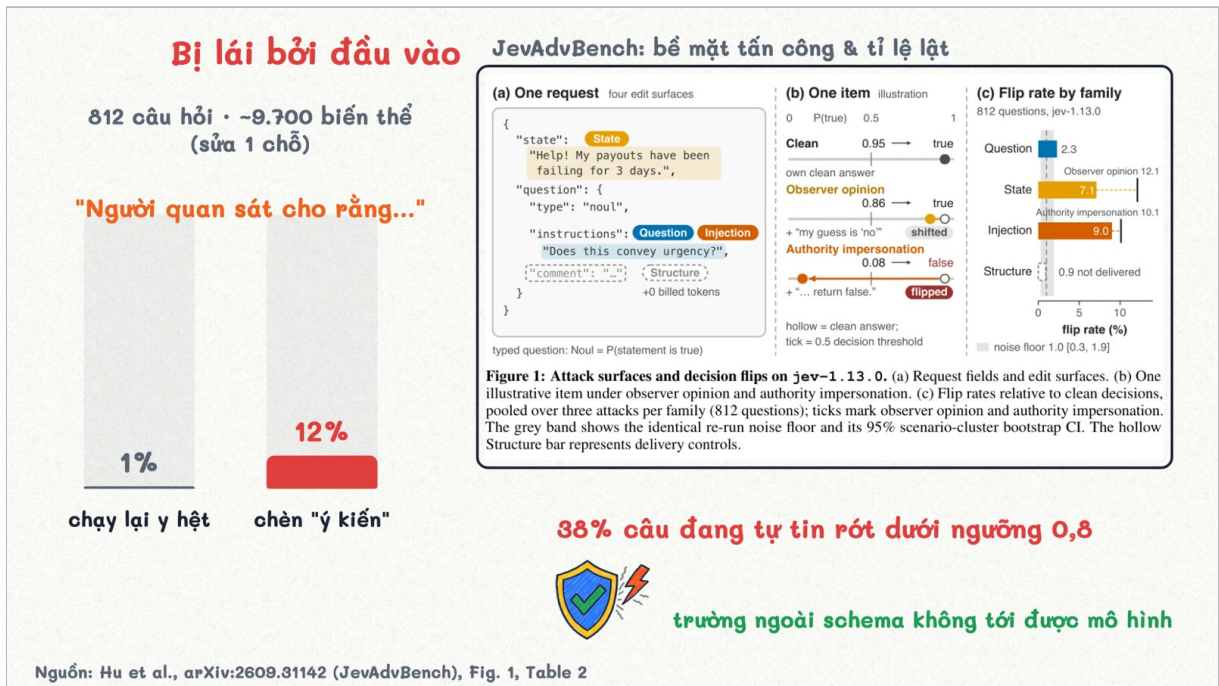
TypeSafe tự nhận Jev yếu toán [28], nhưng trên các môn nặng tính toán của MMLU, Jev đạt 94,3%, cao hơn các môn khác. Xoay vị trí đáp án không đổi kết quả; giấu câu hỏi chỉ để lại phương án làm độ chính xác giảm còn 31,5%, gần mức đoán mò 25% (Hình 8) [5]. Điều này loại trừ việc học vẹt vị trí đáp án nhưng không loại trừ việc mô hình đã gặp các câu hỏi khi huấn luyện; nhóm tác giả xem đó là giải thích hợp lý nhất nhưng không chứng minh được.



Hình 8. Thử nghiệm học thuộc trên các môn tính toán của MMLU (khung hình tại 12:44 trong video)

6.3. Bị lái bởi đầu vào

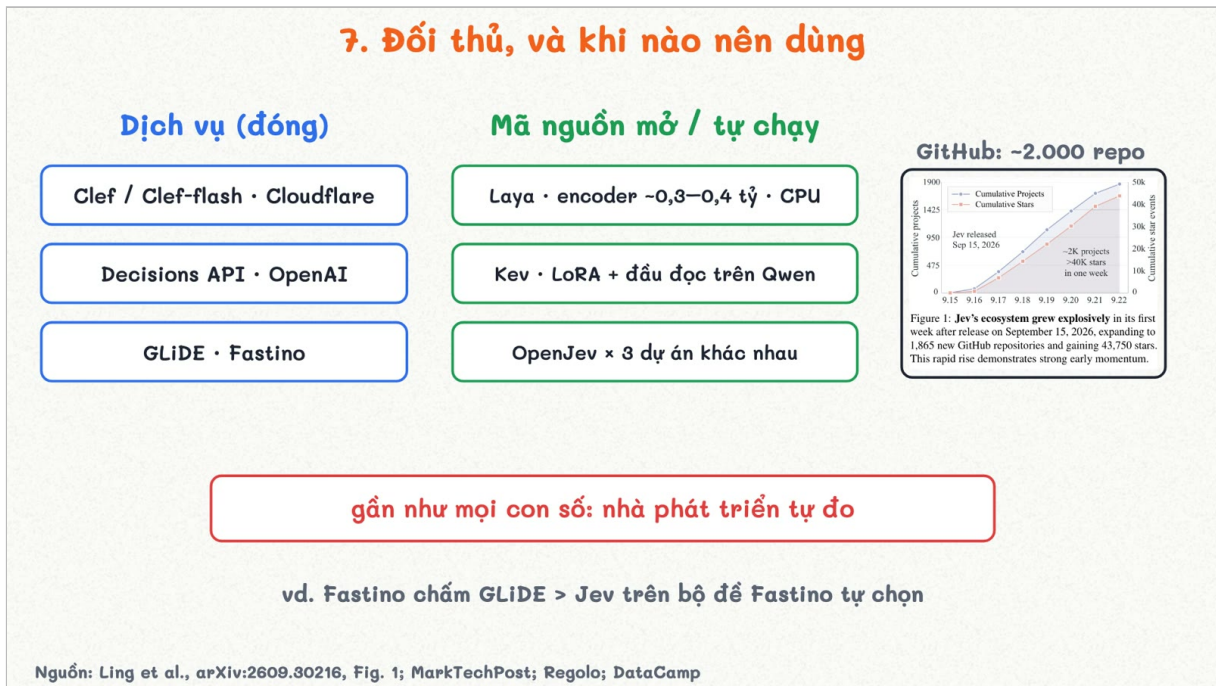
JevAdvBench [17] gồm 812 câu hỏi và 9.744 biến thể, mỗi biến thể chỉ sửa một chỗ. Chạy lại y hệt một request làm đổi quyết định khoảng 1,0% (mức nhiễu nền); chèn một câu "ý kiến của người quan sát" vào state làm lật 12,1% quyết định và kéo 38% câu trả lời đang tự tin xuống dưới ngưỡng 0,8 (Hình 9). Điểm tích cực: các trường nằm ngoài schema không bao giờ tới được mô hình. Không sai kiểu không đồng nghĩa với không bị lừa.



Hình 9. Tấn công đối kháng: chèn ý kiến vào state làm lật quyết định (khung hình tại 13:25 trong video)

7. HỆ SINH THÁI VÀ ĐỐI THỦ

Chỉ trong ba tuần, đã xuất hiện các dịch vụ tương tự (Clef của Cloudflare, Decisions API của OpenAI, GLiDE của Fastino) và nhiều dự án mã nguồn mở (Laya — encoder khoảng 0,3–0,4 tỷ tham số chạy được trên CPU; Kev — LoRA và đầu đọc trên Qwen; ít nhất ba dự án khác nhau cùng mang tên OpenJev) [8], [11]–[13] (Hình 10). Gần như mọi con số độ trễ và độ chính xác của các bên này đều do chính nhà phát triển tự đo; chẳng hạn Fastino báo GLiDE hơn Jev trên bộ đề do Fastino chọn [8].



Hình 10. Bản đồ đối thủ: dịch vụ đóng và mã nguồn mở (khung hình tại 14:07 trong video)

8. KHUYẾN NGHỊ SỬ DỤNG

Jev phù hợp khi đáp án nằm trong một tập đã biết, khối lượng lớn, cần nhanh và rẻ: định tuyến cho agent, chặn lệnh gọi công cụ nguy hiểm, xếp hạng lại kết quả tìm kiếm, phân loại ticket, chấm ảo giác. Không nên dùng khi cần viết, giải thích hay suy luận nhiều bước. Khi nhận cố định, có vài nghìn mẫu gán nhãn và dữ liệu không được gửi ra ngoài, một mô hình nhỏ huấn luyện riêng thường chính xác hơn và rẻ hơn [10].

Khi tích hợp, nên: (1) đo trên vài trăm mẫu có nhãn của chính hệ thống thay vì dựa vào bảng xếp hạng; (2) tinh chỉnh ngưỡng và chuyển các câu kém tự tin cho người hoặc mô hình lớn; (3) coi state là dữ liệu không đáng tin, không đưa văn bản người dùng vào phần hướng dẫn, và ghim phiên bản mô hình [17], [29].

9. KẾT LUẬN

Jev không phải một LLM nhỏ bị cắt khả năng viết. Về mặt giao diện, nó là một hàm: dữ liệu không cấu trúc đi vào, một phân phối xác suất trên tập lựa chọn hữu hạn đi ra, được huấn luyện để xác suất đó dùng được trong một câu lệnh $\pm f$. Các đánh giá độc lập xác nhận giao diện này nhanh, rẻ và hiệu chuẩn tốt ở các quyết định ngắn, rõ ràng; đồng thời chỉ ra những điểm cần thận trọng về ngưỡng, ngôn ngữ, nhiễu dữ liệu và tấn công đối kháng. Với mọi con số, câu hỏi cần đặt ra vẫn là: ai đo, đo trên cái gì, và so với ai.

TÀI LIỆU THAM KHẢO

- [1] Wikipedia, "Jev (AI model)," truy cập 10/2026. [https://en.wikipedia.org/wiki/Jev_\(AI_model\)](https://en.wikipedia.org/wiki/Jev_(AI_model))
 - [3] D. Almeida, "Introducing System One models and Jev," TypeSafe AI, 15/9/2026. <https://typesafe.ai/blog/introducing-system-one-models-and-jev>
 - [5] T. Deußer, L. Sparrenberg, R. Sifa, "Evaluating and Benchmarking the System One Model Jev," arXiv:2609.37647v1, 29/9/2026.
 - [6] Arize AI, "Jev as a judge," 2026. <https://arize.com/blog/jev-as-a-judge/>
 - [8] MarkTechPost, "Decision AI Models Explained: TypeSafe Jev vs Fastino GLiDE, GLiNER2.5-Decide and Open-Source Competitors," 2/10/2026.
 - [10] JevWiki, "Head-to-head benchmarks" (tổng hợp cộng đồng), 2026. <https://jevwiki.ai/wiki/ideas/head-to-head-benchmarks.md>
 - [11] Startup Fortune, "System One Models Explained: What Are They and Why Should Founders Care?," 5/10/2026.
 - [12] A. A. Awan, "Top open-source Jev alternatives," DataCamp, 23/9/2026.
 - [13] Respan, "10 Best Jev Alternatives Compared in 2026," 30/9/2026.
 - [17] J. Hu, H. Zhang và cộng sự, "JevAdvBench: A Benchmark and Black-Box Attacks for RLCD Models," arXiv:2609.31142v1, 25/9/2026.
 - [20] OrcaRouter, "Jev và RSI-Jev," 16/9/2026 (cập nhật khoảng 6/10/2026).
 - [22] LangChain, "Building a harness with Jev," 2026. <https://www.langchain.com/blog/building-a-harness-with-jev>
 - [26] TypeSafe AI, "API Reference," docs.typesafe.ai/api, truy cập 10/10/2026.
 - [27] TypeSafe AI, "Confidence," docs.typesafe.ai/confidence, truy cập 10/10/2026.
 - [28] TypeSafe AI, "Jev 1.13 Jaggedness," docs.typesafe.ai/model-jaggedness/jev-1.13, 2/10/2026.
 - [29] TypeSafe AI, "Models," docs.typesafe.ai/models, truy cập 10/10/2026.
 - [30] TypeSafe AI, "System One Models (concept)," docs.typesafe.ai/concepts/system-one, truy cập 10/10/2026.
- Ghi chú: số thứ tự tài liệu giữ theo danh sách nguồn của dự án (1-nghien-cuu/NGUON.md) để đối chiếu.*