خوب ساختن با هوش مصنوعی

راهنمای گام‌به‌گامِ نرم‌افزار، عامل‌ها و ارکستراسیون‌ها — به ترتیبی که نتایجِ بهتری می‌دهد

Rehm KI Consulting

نسخهٔ ۱٫۰ · ویرایش نخست · ژوئیهٔ ۲۰۲۶

چرا ترتیب همان متغیرِ پنهان است

وقتی یک ساختِ به‌کمکِ هوش مصنوعی خراب می‌شود، نخستین واکنش این است که مدل را مقصر بدانیم — به‌اندازهٔ کافی باهوش نبود، توهم زد، کدِ شلخته نوشت. معمولاً این علتِ واقعی نیست. علتِ واقعی تقریباً همیشه **ترتیب** است: کار به توالیِ نادرست انجام شد. پیش از تعریفِ مسئله کد نوشته شد. از مدل خواسته شد که به‌جای گرفتنِ مواد، به یاد آورد. همان چیزی که سیستم را ساخت، تنها چیزی هم بود که آن را بررسی کرد. کاربرانِ واقعی نخستین آزمون‌گرها بودند.

ترتیب را درست کنید، یک مدلِ معمولی کارِ خوب تولید می‌کند. آن را نادرست کنید، بهترین مدلِ جهان یک آشفتگیِ پرمدعا تولید می‌کند.

این راهنما همان ترتیب را شرح می‌دهد. این ترتیب برای سه نوع کار به کار می‌آید، که کمتر از آنچه به نظر می‌رسد با هم فرق دارند:

- **نرم‌افزارِ ساده** — یک برنامه یا سرویسِ معمولی.
- **عامل‌های هوش مصنوعی** — سیستم‌هایی که در آن‌ها یک مدل از راهِ ابزارها و با اندکی خودمختاری دست به کنش می‌زند.
- **ارکستراسیون‌ها و روال‌ها** — چند عامل یا گام که در یک خط لوله ترکیب شده‌اند، جایی که خودِ هماهنگی همان محصول است.

گام‌ها برای هر سه یکسان‌اند. هرجا که یک عامل یا یک ارکستراسیون به چیزِ افزون‌تری نیاز داشته باشد، همان گام آن را می‌گوید.

سه اندیشه زیرِ همهٔ آنچه در پی می‌آید جاری‌اند و ارزش دارد پیش از آغازِ گام‌ها روشن بیان شوند. **پیش از ساختن، تعریف کنید** — درخواستِ بیان‌شده به‌ندرت همان مسئلهٔ واقعی است. **لنگر بیندازید، نه یادآوری** — هوش مصنوعی وقتی معتمد است که مواد پیش رویش باشد و وقتی نامطمئن است که دست به حافظه می‌برد. **سازنده را از بازبین جدا کنید** — هیچ چیزی که چیزی را ساخته نباید داورِ یگانهٔ خوب‌بودنش باشد. تقریباً هر گامِ زیر یکی از این سه اندیشه است که در لحظه‌ای دیگر به کار بسته شده.

گام ۰ — تعریفِ مسئلهٔ واقعی

چه کنید: ایده را به یک سند ایدهٔ واحد و منسجم برسانید. در آن، هدف، نیت‌های خود و — این همان بخشی است که مردم از آن می‌گذرند — نمونه‌های عینی از اینکه **موفقیت** چه شکلی است و **شکست** چه شکلی است را بیاورید.

چرا مهم است: درخواستِ بیان‌شده یک نقطهٔ آغاز است، نه یک مشخصات. مردم نشانه‌ها را وصف می‌کنند نه علت‌ها را؛ یک راه‌حلِ فرض‌شده را وصف می‌کنند نه نیازِ زیرین را. اگر دقیقاً همان چیزی را بسازید که خواسته شد، اغلب چیزِ نادرست را زیبا می‌سازید. نمونه‌های شکست به‌اندازهٔ نمونه‌های موفقیت اهمیت دارند، چون مرزی را می‌کشند که سیستم نباید از آن بگذرد.

نشانهٔ کار خوب: یک صفحه که یک غریبه می‌تواند بخواند و درست توضیح دهد چه می‌سازید، برای چه کسی، و چگونه هر دوی شما می‌فهمید که کار گرفته است.

با ساده‌ترین نسخهٔ مفید شروع کنید. در برابرِ وسوسهٔ تعریفِ کلِ سیستمِ بلندپروازانه به‌یک‌باره مقاومت کنید. کوچک‌ترین مسئله‌ای را که ارزشِ واقعی می‌دهد برگزینید، آن را واقعاً خوب به کار بیندازید، و توانایی‌ها را در تکرارهای بعدی (گام ۶) بیفزایید. یک چیزِ سادهٔ کارآمد بر یک چیزِ پیچیدهٔ نیمه‌کاره می‌چربد — و نسخهٔ ساده به شما می‌آموزد که نسخهٔ پیچیده واقعاً به چه نیاز دارد.

خطای رایج: معیارهای موفقیتِ مبهم («خوبش کن»، «کاربران باید دوستش داشته باشند»). اگر موفقیت عینی نباشد، نمی‌توانید آن را بیازمایید و نمی‌توانید بگویید کِی تمام شده‌اید.

- **برای عامل‌ها:** تعریف کنید که عامل مسئولِ چیست و، به همان اندازه مهم، چه چیزی را هرگز نباید انجام دهد. «موفقیت» و «شکست» به رفتارهای مشخص بدل می‌شوند — کنش‌هایی که می‌خواهید و کنش‌هایی که فاجعه خواهند بود.
- **برای ارکستراسیون‌ها:** پیش از تجزیه، نخست نتیجهٔ سرتاسری را تعریف کنید. یک خط لوله از گام‌های به‌تنهایی درست که در مجموع به هدف نمی‌رسند، خطایی کلاسیک و پرهزینه است.

گام ۱ — گردآوریِ بستهٔ زمینه

چه کنید: هر آنچه را که هوش مصنوعی باید از رویِ آن بسازد در قالب اسنادِ واقعی گرد آورید: سند ایده، هدف، نیت‌ها، فهرست ویژگی‌ها، نمونه‌های موفقیت و شکست، ارزش‌ها و محدودیت‌ها، و استانداردهای کدنویسی. از این‌ها یک سند طراحیِ کوتاه بسازید.

چرا مهم است: این بزرگ‌ترین اهرمِ کیفیت و قوی‌ترین دفاع در برابر توهم است. یک مدلِ زبانی متنِ محتمل را پیش‌بینی می‌کند؛ وقتی فاکتی کم دارد، فاکتی محتمل می‌سازد، با همان صدایِ پرمدعایی که برای دانسته‌هایش به کار می‌برد. راه‌حل امید بستن به یادآوریِ آن نیست — این است که حقیقت را پیشِ رویش

بگذارید. ساختی که در یک بستهٔ زمینهٔ فراهم‌شده از سوی شما لنگر انداخته، از رویِ واقعیتِ شما کار می‌کند، نه از رویِ تقریبِ مدل.

نشانهٔ کار خوب: پوشه‌ای از اسناد روشن. هیچ چیزِ باربَری فقط در سرِ شما یا در تاریخچهٔ گفت‌وگویی که هفتهٔ بعد از دستش می‌دهید زندگی نمی‌کند.

خطای رایج: محدودیت‌های تعیین‌کننده (یک الزامِ مجوز، یک قاعدهٔ امنیتی، یک قالبِ داده) که «بدیهی» بودند و بنابراین هرگز نوشته نشدند — پس مدل هرگز از آن‌ها خبر نداشت.

- **برای عامل‌ها:** بستهٔ زمینه به system prompt عامل، تعریف ابزارهایش، حفاظ‌هایش و مجموعه‌ای از نمونه‌های کارشده بدل می‌شود. دربارهٔ ابزارهای در دسترس و مرزهای تثبیت‌شده صریح باشید.
- **برای ارکستراسیون‌ها:** زمینهٔ مشترک و قراردادها میان مؤلفه‌ها را تعریف کنید — اینکه هر کدام دربارهٔ دیگری چه می‌تواند فرض کند. ابهام میان عامل‌ها همان‌جایی است که ارکستراسیون‌ها می‌پوسند.

گام ۲ — نوشتنِ مشخصات

چه کنید: بگذارید یک مدلِ استدلالیِ قوی از رویِ بستهٔ زمینه یک سند مشخصاتِ نوشتاری تهیه کند. سپس شما آن را بخوانید، اصلاح کنید و تأیید کنید. تأییدِ انسانی اختیاری نیست.

چرا مهم است: مشخصات همان قرارداد میان نیت و پیاده‌سازی است. خطایی در مشخصات اکنون ارزان است و به‌محض آنکه بر آن ساخته شود گران می‌گردد. بازبینیِ مشخصات پرثمرترین نیم‌ساعتِ کلِ پروژه است.

نشانهٔ کار خوب: مشخصاتی که شخصاً خوانده‌اید و واقعاً با آن موافقید — آن‌قدر دقیق که ابهام را بردارد، اما نه آن‌قدر خشک که هر انتخابِ پیاده‌سازی را دیکته کند و بهترها را ببندد.

خطای رایج: سرسری خواندنِ مشخصات چون «درست به نظر می‌رسد». مشخصاتی که درست به نظر می‌رسد و به‌طورِ ظریف نادرست است، از نبودِ مشخصات بدتر است، چون اکنون همه به آن اعتماد می‌کنند.

- **برای عامل‌ها:** نقش، ورودی‌ها و خروجی‌ها، ابزارها و شرایط توقف را مشخص کنید — اینکه عامل کِی باید تحویل دهد، ارجاع دهد یا رد کند.
- **برای ارکستراسیون‌ها:** هر تحویل‌دهی را مشخص کنید. پیام‌های میان گام‌ها همان رابطِ واقعی‌اند؛ آن‌ها را با همان دقتی مشخص کنید که یک API عمومی را.

گام ۳ — نوشتنِ یک طرحِ ساختِ مفصل

چه کنید: از رویِ مشخصاتِ تأییدشده، یک طرحِ ساختِ مفصل و مرتب تولید کنید — آن‌قدر مفصل که یک مدلِ کوچک‌تر و ارزان‌تر بتواند آن را بی‌آنکه به استدلالی که تولیدش کرده نیاز داشته باشد، تا پایان اجرا کند.

چرا مهم است: یک طرحِ قوی همان چیزی است که به شما اجازه می‌دهد بخشِ عمدهٔ کار را به مدل‌های اقتصادی بسپارید و هزینه‌ها را معقول نگه دارید (به «مدلِ درست برای هر کار» نگاه کنید). همچنین ماژولاریت را الزامی می‌کند: برای نوشتنِ طرح باید درزهایی را بیابید که سیستم در آن‌ها به‌طور طبیعی جدا می‌شود، و این درزها به رابط‌های تمیز بدل می‌شوند نه وابستگی‌های درهم‌تنیده.

نشانهٔ کار خوب: کارهای مجزا و مرتب با رابط‌های روشن میانشان — هرکدام یا مستقیماً ساختنی یا مستقیماً بازبینی‌شدنی.

خطای رایج: طرحی که در حقیقت یک مونولیتِ مبدل است — گام‌هایی چنان درهم‌تنیده که هیچ‌چیز را نمی‌توان جداگانه ساخت یا آزمود.

- **برای عامل‌ها و ارکستراسیون‌ها:** قراردادهای پیام، مسیریابی (چه کسی چه کسی را و کِی فرا می‌خواند) و مدیریت خطا را پیش از ساختن تعریف کنید. از پیش تصمیم بگیرید وقتی گامی زمانش سر می‌آید، آشغال برمی‌گرداند یا در حلقه می‌افتد چه رخ می‌دهد. ماژولار بسازید، با یک گذرگاه ارتباطی میان مؤلفه‌ها به‌صورت پیش‌فرض، مگر آنکه دلیلِ مشخصی خلافش باشد.

گام ۴ — ساختن با انضباط

چه کنید: طرح را اجرا کنید. از همان خط نخست، پادشکننده و امن بسازید، نه به‌عنوان یک مرحلهٔ بعدی. آزمون‌محور و ماژولار کار کنید. آزمون‌ها را حین ساختِ سیستم پیوسته اجرا کنید و بخواهید که پیش از هر push هر آزمون به‌صورت محلی پاس شود.

چرا مهم است: امنیت و تاب‌آوری‌ای که در پایان وصله شوند هرگز درست جا نمی‌افتند — فرض‌ها پیش‌تر پخته شده‌اند. پادشکننده یعنی سیستم چنان طراحی شده که وقتی بخشی از کار می‌افتد، بقیه به کار ادامه می‌دهند؛ خطا جدا می‌شود نه آنکه آبشاری شود. این تفاوتِ میان یک بعدازظهرِ بد و یک فاجعه است، و برای هر چیزی که به‌عنوان سرویس اجرا می‌شود، اختیاری نیست.

نشانهٔ کار خوب: یک ساختِ ماژولار، آزمون‌ها به‌صورت محلی سبز، و افتِ کنترل‌شدهٔ نمایش‌پذیر — می‌توانید یک مؤلفه را بکُشید و ببینید بقیه پابرجا می‌مانند.

خطای رایج: «امنیت و آزمون‌ها را بعداً اضافه می‌کنیم.» بعداً نمی‌آید، و وقتی بیاید، یک بازنویسی است.

- **برای عامل‌ها:** حفاظ‌ها، اعتبارسنجیِ ورودی، دسترسیِ ابزارِ محصورشده (sandbox) و زمان‌بندی‌ها و تلاش‌های مجددِ معقول. فرض کنید مدل گاهی خروجیِ بد تولید می‌کند و چنان بسازید که خروجیِ بد نتواند آسیب بزند.
- **برای ارکستراسیون‌ها:** دامنه‌های خطا را جدا کنید تا یک عاملِ ازکارافتاده خط لوله را مسموم نکند. گام‌ها را هرجا می‌توانید ایدمپوتنت (idempotent) کنید و مدارشکن (circuit breaker) بیفزایید تا یک مؤلفهٔ گیرکرده یا در حلقه‌افتاده متوقف شود نه آنکه به کار رها گردد.

گام ۵ — بازبینیِ مستقل و سخت‌سازی

چه کنید: بگذارید **دو نشستِ جداگانهٔ** هوش مصنوعی به‌صورت خصمانه کار را نقد کنند. آن‌ها فقط نقد می‌کنند — چیزی را اصلاح نمی‌کنند. یافته‌هایشان برای پیاده‌سازی به سازندهٔ اصلی بازمی‌گردد. سپس تستِ نفوذ را چند بار اجرا کنید، نه یک‌بار.

چرا مهم است: حسابدار را نمی‌گذارید دفترهای خودش را حسابرسی کند، به همان دلیل که نباید عاملی را که کد را نوشته تنها داورِ آن قرار دهید: تضاد منافع ساختاری است، نه مسئلهٔ حُسنِ نیت. بازبینی با زمینهٔ تازه و بی‌سهم در کار، آنچه را که نویسنده ذاتاً نسبت به آن کور است می‌بیند. جدا نگه‌داشتنِ نقد از اصلاح این استقلال را حفظ می‌کند — لحظه‌ای که یک منتقد به اصلاح دست می‌زند، سهمی به دست می‌آورد.

مواردِ غیرقابل‌مذاکره. سه چیز اینجا مسئلهٔ صلاحدید نیستند. نخست، همه‌چیز از یک خط لولهٔ CI/CD می‌گذرد — هر تغییر، هر بار، به‌صورت خودکار ساخته و آزموده می‌شود. دوم، همهٔ آزمون‌ها همیشه پاس می‌شوند؛ یک ساختِ قرمز هرگز merge نمی‌شود و هرگز «برای اصلاح در آینده» تحویل داده نمی‌شود. سوم، تستِ نفوذ الزامی است، نه تجملی برای وقتی که فرصت هست — سیستمی که تستِ نفوذ نشده، سیستمی نامعتمد است.

نشانهٔ کار خوب: نقدهای مشخص و اجراشدنی که به‌راستی اجرا می‌شوند؛ سازنده و بازبینی که هرگز یک عامل نیستند؛ و تست‌های نفوذی که تکرار شده‌اند تا دیگر مسئلهٔ مهمی به بار نیاورند.

خطای رایج: یک گذرِ بازبینیِ واحد، یا «بازبینی‌ای» که در حقیقت همان نشست با کلاهی دیگر است. استقلالی که واقعاً الزامش نکرده‌اید، استقلالی است که ندارید.

- **برای عامل‌ها و ارکستراسیون‌ها:** پرامپت‌ها و جریان را به‌صورت خصمانه بیازمایید — تزریق پرامپت (Prompt Injection)، جیل‌بریک، ورودی‌های ناقص، پاسخ‌های خصمانهٔ ابزار. کلِ ارکستراسیون را تیمِ قرمز (Red-Teaming) کنید، نه فقط عامل‌های منفرد را؛ خطاهای جالب در تحویل‌دهی‌ها زندگی می‌کنند.

گام ۶ — آزمونِ انسانی، سپس تکرار

چه کنید: نخستین آزمونِ انسانی را زمانی اجرا کنید که سیستم به‌طور کامل بالا و در حال اجراست — نه پیش از آن. از آنجا به بعد، بر پایهٔ بازخوردِ واقعی به‌صورت تکراری توسعه دهید.

چرا مهم است: آزمودنِ یک سیستمِ نیمه‌ساخته توجهِ انسانی را صرفِ مسائلی می‌کند که خودِ ساخت خودبه‌خود حلشان می‌کرد. داوریِ انسانی کمیاب و ارزشمند است؛ آن را به‌سوی یک چیزِ واقعیِ در حال اجرا نشانه بگیرید، جایی که بازخورد واقعی است.

نشانهٔ کار خوب: یک سیستمِ در حال اجرا در برابر یک انسان، که فهرستِ کوتاه و اولویت‌بندی‌شده‌ای از بهبودهای واقعی تولید می‌کند — نه تقلایی برای به‌راه‌انداختنش.

خطای رایج: استفاده از کاربرانِ واقعی به‌عنوان نخستین آزمونِ یکپارچگی. آن‌ها درمی‌یابند که خراب است، نه اینکه ناقص است، و اعتماد را سوزانده‌اید تا چیزی را بیاموزید که یک آزمونِ دود (smoke test) به شما می‌گفت.

مدلِ درست برای هر کار

دشوارترین تفکر — تعریف، مشخصات، برنامه‌ریزی و نقد — را به قوی‌ترین مدلی بسپارید که به آن دسترسی دارید. اجرای روزمره را به ارزان‌ترین مدلی بسپارید که کار را قابل‌اعتماد انجام می‌دهد. استفاده از یک مدلِ گران برای همه‌چیز پول را هدر می‌دهد؛ استفاده از یک مدلِ ارزان برای استدلالِ دشوار همه‌چیز را هدر می‌دهد، چون خطاها به هر گامِ پایین‌دستی تکثیر می‌شوند.

یک رده‌بندیِ عملی، **که صادقانه به‌عنوان یک وضعیتِ کاری که زود کهنه می‌شود بیان می‌شود** — نام‌ها و نسخه‌های مدل‌ها پیوسته تغییر می‌کنند و هر رتبه‌بندیِ مشخص باید پیش از تکیه بر آن دوباره سنجیده شود:

- **ردهٔ برتر — دشوارترین استدلال:** تعریف، مشخصات، طرحِ ساخت، نقدِ خصمانه.
- **ردهٔ میانیِ توانمند و مدل‌های اقتصادی — اجرای روزمره:** پیاده‌سازیِ کارهای خوب‌مشخص‌شده از رویِ یک طرحِ مفصل.

انضباط، خودِ فهرستِ مشخص نیست؛ عادتِ تطبیقِ توانایی با دشواریِ کار و سنجشِ دوره‌ایِ دوباره است، به‌جای اعتماد به رتبه‌بندی‌ای که شش‌ماه کهنه شده.

سازنده را از بازبین جدا کنید

این قلبِ ارکستراسیونِ خوب است و از این رو بخشِ ویژهٔ خود را می‌گیرد. اصل همان اصلِ حسابدار است: هرکه کار را انجام می‌دهد نباید تنها کسی باشد که آن را بررسی می‌کند. وقتی تصمیم می‌گیرید کار چگونه میان مدل‌ها یا عامل‌ها تقسیم و واگذار شود، هر کار را از چهار پرسش بگذرانید:

1. **اندازه** — کار برای یک نشستِ واحد چند توکن است؟ کارهایی که به‌تمیزی در یک پنجرهٔ زمینه جا نمی‌شوند نیاز به تقسیم یا ساختاری دیگر دارند.
2. **استقلال** — آیا می‌توان کار را بدون از دست دادنِ اثربخشی تقسیم کرد؟ برخی چیزها تمیز تجزیه می‌شوند؛ برخی معنایشان را از دست می‌دهند وقتی بریده شوند. در امتدادِ درزهای طبیعی تقسیم کنید، نه دلبخواهی.
3. **تفکیک وظایف** — آیا انجامِ هر دو نیمه توسط یک عامل تضاد منافع می‌سازد؟ اگر چنین است، جدایشان کنید، حتی به بهای اندکی کارایی.
4. **بازبینی‌پذیری** — آیا بررسیِ کار بسیار ارزان‌تر از انجامِ آن است؟ اگر چنین است، از چند عاملِ بازبینِ مستقل استفاده کنید. تأییدِ ارزان که به‌موازات اجرا می‌شود، اعتمادِ فراوانی به بهای اندک می‌خرد.

آن نکتهٔ آخر ابرقدرتِ خاموشِ ارکستراسیون است: تأیید اغلب بسیار ارزان‌تر از تولید است، پس می‌توانید از پسِ آن برآیید که همان خروجی را چند بار به شیوه‌های مختلف بررسی کنید و چیزی را بگیرید که هر بررسیِ منفرد از دست می‌داد.

ضدالگوها، یک‌جا

- **حل مسئلهٔ بیان‌شده به‌جای مسئلهٔ واقعی.** نخست تعریف کنید.
- **گذاشتنِ مدل به یادآوری به‌جای دادنِ مواد به آن.** همه‌چیز را لنگر بیندازید.
- **نوشتنِ کد پیش از وجودِ مشخصات و طرح.** ترتیب همان کلِ بازی است.
- **مونولیتِ مبدل** — «طرحی» که گام‌هایش را نمی‌توان جداگانه ساخت یا آزمود.
- **امنیت و آزمون‌ها «بعداً».** بعداً یک بازنویسی است.
- **خودبازبینی** — سازنده کارِ خودش را نمره می‌دهد.
- **یک تستِ نفوذ.** تکرار کنید تا خاموش شود.
- **کاربران به‌عنوان نخستین آزمون‌گرها.** آزمونِ انسانی روی یک سیستمِ کاملاً در حال اجرا آغاز می‌شود.
- **یک مدل برای همه‌چیز** — یا برای کارِ روزمره گران می‌پردازید یا استدلالِ دشوار را کم‌توان می‌گذارید.

نتیجهٔ صادقانه

هیچ‌کدامِ این‌ها عجیب نیست. هر گام به‌محضِ بیان‌شدن بدیهی است. دشواری این است که زیرِ فشارِ اینکه فقط شروع به ساختن کنیم، واکنش‌های پیش‌فرض بازمی‌گردند — از تعریف بگذر، به حافظهٔ مدل اعتماد کن، بگذار خودش را بررسی کند، به کاربران تحویل بده «تا ببینیم چه می‌شود». خوب انجام دادنِ این کار بیش از هر چیز انضباطِ پیرویِ از ترتیب است، حتی وقتی جلو پریدن سریع‌تر به نظر می‌رسد. سریع‌تر نیست. فقط این‌طور حس می‌شود — تا گام ۶.

این یک راهنمای کلی و فراگیر دربارهٔ خوب ساختن با هوش مصنوعی است. آگاهانه در سطحِ اصول و ترتیب می‌ماند و نمی‌تواند جزئیاتِ خاصِ یک پروژهٔ منفرد را پوشش دهد. برای راهنماییِ مفصل و متناسب با موقعیتِ شما، تماس بگیرید.

دربارهٔ Rehm KI Consulting

Rehm KI Consulting به سازمان‌ها کمک می‌کند با هوش مصنوعی همان‌گونه بسازند که باید ساخته شود — با انضباطِ مهندسیِ واقعی در پسِ آن، نه یک دموی آراسته به‌عنوان محصول. این رویکرد در دهه‌ها تجربهٔ سخت‌افزار و نرم‌افزار ریشه دارد، از طراحیِ تراشه و پردازنده‌های شبکه در Intel تا سال‌ها ساختِ برنامه‌های وب، و امروز بر کمک به کسب‌وکارها برای به‌کارگیریِ هوش مصنوعی متمرکز است — صادقانه، قابل‌اعتماد و ساخته‌شده برای ماندگاری.

موضع ساده است: پاسخ‌های صریح دربارهٔ اینکه هوش مصنوعی واقعاً کجا کمک می‌کند و کجا نه. بدون هیاهو، بدون جادو.

Rehm KI Consulting · Christopher Rehm · بایرن، آلمان · contact@christopherrehm.de

نام‌ها، ابزارها و رتبه‌بندی‌های نسخهٔ مدل‌هایی که در این راهنما ذکر شده‌اند به‌سرعت تغییر می‌کنند — آن‌ها را نمونه‌هایی برای بازبینیِ دوباره بدانید، نه فاکت‌های ثابت.