کتابچهٔ عملیاتیِ ساخت با هوش مصنوعی

ساخت نرم‌افزار، عامل‌ها و ارکستراسیون‌ها به ترتیب درست — و رسیدن به نتایجی به‌مراتب بهتر

یک سیاههٔ کاربردی از Rehm KI Consulting

نسخهٔ ۱٫۰ · ویرایش نخست · ژوئیهٔ ۲۰۲۶

بیشتر نتایج ضعیف در ساخت به‌کمکِ هوش مصنوعی از یک مدلِ ضعیف نمی‌آید. آن‌ها از انجام گام‌ها به ترتیب نادرست پدید می‌آیند — پریدن به کدنویسی پیش از آنکه مسئله روشن تعریف شود؛ اجازه دادن به یک هوش مصنوعی که کارِ خودش را ارزیابی کند؛ آزمودن روی کاربران واقعی به‌جای آزمودن پیش از آن‌ها.

این همان ترتیبی است که جواب می‌دهد. آن را از بالا به پایین دنبال کنید. هر گام می‌گوید **چه** باید کرد، **چرا** اهمیت دارد، و **«کارِ خوب» چه شکلی است** تا بدانید کِی می‌توانید جلو بروید.

> **چگونه این را بخوانید:** متنِ اصلی برای هرکسی که یک پروژهٔ ساختِ هوش مصنوعی را هدایت می‌کند کارآمد است. خط‌هایی که با **[توسعه‌دهنده]** نشان‌گذاری شده‌اند جزئیات فنی را برای برنامه‌نویسان می‌افزایند. اگر برای شما نیست، از آن‌ها بگذرید.

گام ۰ — تعریفِ مسئلهٔ واقعی

چه کنید: پیش از هر چیز، ایده را به یک سند ایدهٔ واحد و منسجم برسانید. هدف، نیت‌های خود و نمونه‌های عینی از اینکه **موفقیت** و **شکست** چه شکلی‌اند را بنویسید.

چرا مهم است: درخواستِ بیان‌شده یک نقطهٔ آغاز است، نه یک مشخصات. بیشترِ پروژه‌های هدررفته مسئلهٔ نادرست را حل می‌کنند — فقط خوب حل می‌کنند. یک ساعت صرف‌شده در اینجا روزها را بعداً نجات می‌دهد.

نشانهٔ کار خوب: یک صفحه که می‌توانید به یک غریبه بدهید و او بفهمد چه می‌سازید، برای چه کسی، و چگونه می‌فهمید که کار گرفته است.

کوچک شروع کنید: نخست ساده‌ترین نسخهٔ مفیدِ مسئله را برگزینید. همان یک چیز را واقعاً خوب به کار بیندازید و بعداً در تکرارها (گام ۶) به آن بیفزایید. نکوشید همه‌چیز را یک‌جا بسازید — یک چیزِ سادهٔ کارآمد بر یک چیزِ پیچیدهٔ نیمه‌کاره می‌چربد.

گام ۱ — گردآوریِ بستهٔ زمینه

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

چرا مهم است: هوش مصنوعی وقتی قابل‌اعتماد است که مواد پیش رویش باشد و وقتی نامطمئن است که ناچار از یادآوری از حافظه باشد. یک بستهٔ زمینهٔ خوب، در عین حال، قوی‌ترین دفاع شما در برابر خروجیِ توهمی (hallucination) است — شما کار را در واقعیت‌هایی که خودتان فراهم کرده‌اید لنگر می‌اندازید.

نشانهٔ کار خوب: پوشه‌ای از اسناد روشن. هیچ چیز حیاتی‌ای فقط در سرِ شما یا در تاریخچهٔ گفت‌وگویی که از دستش خواهید داد زندگی نمی‌کند.

> **[توسعه‌دهنده]** انتظارات رابط‌ها، شکلِ داده‌ها و مواردِ غیرقابل‌مذاکره را بگنجانید (وضعیت امنیتی، مجوزها، محیط اجرای هدف). این می‌شود منبعی که مشخصات در برابر آن نوشته می‌شود.

گام ۲ — نوشتنِ مشخصات

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

چرا مهم است: مشخصات همان قرارداد است. خطاهایی که اینجا گرفته شوند ارزان‌اند؛ همان خطاها پس از ساخت چنین نیستند.

نشانهٔ کار خوب: مشخصاتی که خودتان شخصاً خوانده‌اید و واقعاً با آن موافقید — نه مشخصاتی که سرسری از آن گذشته‌اید.

> **[توسعه‌دهنده]** برای این کار از یک مدلِ استدلالیِ ردهٔ نخست استفاده کنید (به «مدلِ درست برای هر کار» نگاه کنید). مشخصات باید آن‌قدر دقیق باشد که ابهام را بردارد، اما نه آن‌قدر خشک که انتخاب‌های خوبِ پیاده‌سازی را ببندد.

گام ۳ — نوشتنِ یک طرحِ ساختِ مفصل

چه کنید: از رویِ مشخصات، یک طرحِ ساختِ مفصل و گام‌به‌گام تولید کنید — آن‌قدر مفصل که یک مدلِ کوچک‌تر و ارزان‌تر بتواند آن را تا پایان اجرا کند.

چرا مهم است: یک طرحِ قوی به شما اجازه می‌دهد بارِ سنگین را به مدل‌های اقتصادی بسپارید و ساخت را ماژولار نگه دارید نه درهم‌تنیده.

نشانهٔ کار خوب: کارهای مجزا و مرتب با رابط‌های روشن میانشان. کسی (یا چیزی) کم‌توان‌تر از طراح بتواند از آن پیروی کند.

> **[توسعه‌دهنده]** ماژولار بسازید، با یک گذرگاه ارتباطی (communication bus) میان مؤلفه‌ها به‌صورت پیش‌فرض، مگر آنکه دلیل خوبی خلافش باشد. درزهایی را که سیستم در آن‌ها به‌طور طبیعی جدا می‌شود مشخص کنید.

گام ۴ — ساختن با انضباط

چه کنید: طرح را اجرا کنید. از همان خط نخست، پادشکننده (anti-fragile) و امن بسازید. آزمون‌محور و ماژولار کار کنید. آزمون‌ها را حین ساخت پیوسته اجرا کنید و پیش از هر push، هر آزمون را به‌صورت محلی از سر بگذرانید.

چرا مهم است: امنیت و تاب‌آوری‌ای که در پایان وصله می‌شوند هرگز درست جا نمی‌افتند. طراحیِ پادشکننده یعنی شکستِ یک مؤلفه کلِ سیستم را با خود پایین نمی‌کشد.

نشانهٔ کار خوب: یک ساختِ ماژولار که آزمون‌هایش به‌صورت محلی سبزند، و شکستِ یک بخش به‌جای فروپاشیِ همه‌چیز، به‌نرمی افت می‌کند.

> **[توسعه‌دهنده]** توسعهٔ آزمون‌محور (TDD) در سراسر کار. پادشکنندگی یک الزامِ طراحی است نه یک امکانِ تجملی — دامنه‌های خطا را جدا کنید. با وضعیتِ قرمز، push نکنید.

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 اصلاح به سازندهٔ اصلی بازمی‌گردد. تستِ نفوذ را چند بار اجرا کنید.

چرا مهم است: نگذارید حسابدار دفترهای خودش را حسابرسی کند. بازبینی‌کننده‌ای که سهمی در کار ندارد آنچه را که سازنده نسبت به آن کور است می‌بیند. همین جدایی نکتهٔ اصلی است.

نشانهٔ کار خوب: نقدها مشخص‌اند و به کار بسته می‌شوند؛ سازنده و بازبین هرگز یک عامل نیستند؛ تست‌های نفوذ تکرار شده‌اند، نه یک‌بار اجرا.

> **[توسعه‌دهنده]** برای هر منتقد زمینهٔ تازه — بدون حافظهٔ مشترک با سازنده. حلقهٔ نقد/اصلاح را تکرار کنید تا منتقدان دیگر مسئلهٔ مهمی نیابند.

غیرقابل‌مذاکره: همه‌چیز را از یک خط لولهٔ CI/CD بگذرانید، هر بار — با هر تغییر به‌صورت خودکار ساخته و آزموده شود. همهٔ آزمون‌ها همیشه پاس می‌شوند؛ یک ساختِ قرمز هرگز merge نمی‌شود و هرگز «برای اصلاح در آینده» تحویل داده نمی‌شود. و تستِ نفوذ یک الزامِ مطلق است، نه گامی که هنگام کمبود وقت از آن بگذرید — سیستمی که تستِ نفوذ نشده، سیستمی نامعتمد است.

گام ۶ — آزمونِ انسانی، سپس تکرار

چه کنید: نخستین آزمونِ انسانی زمانی رخ می‌دهد که سیستم به‌طور کامل بالا و در حال اجراست. از آنجا به بعد، به‌صورت تکراری توسعه دهید.

چرا مهم است: آزمودن یک سیستمِ نیمه‌ساخته توجه همه را صرف مسائلی می‌کند که خودِ ساخت به‌هرحال حلشان می‌کرد. بازخوردِ واقعی به یک چیزِ واقعیِ در حال اجرا تعلق دارد.

نشانهٔ کار خوب: یک سیستمِ در حال اجرا در برابر یک انسان، و فهرستِ کوتاهی از بهبودهای بعدی — نه تقلایی برای اینکه اصلاً راه بیفتد.

دو قاعده که در هر گام جاری‌اند

مدلِ درست برای هر کار. دشوارترین تفکر — تعریف، مشخصات، طرح، نقد — را به بهترین مدلِ در دسترس بسپارید. اجرای روزمره را به ارزان‌ترین مدلی بسپارید که کار را خوب انجام می‌دهد. تطبیقِ ابزار با کار هم قابل‌اعتمادتر است و هم ارزان‌تر.

> **[توسعه‌دهنده]** یک رده‌بندیِ کاری که در طول زمان تنظیم می‌شود (نام‌ها و نسخه‌ها تغییر می‌کنند): ردهٔ برتر برای استدلالِ دشوار؛ ردهٔ میانیِ توانمند و مدل‌های اقتصادی برای اجرای روزمره. به‌طور دوره‌ای دوباره سنجش کنید — این فهرست زود کهنه می‌شود.

سازنده را از بازبین جدا کنید. همان تضاد منافعی که حسابداران را از حسابرسیِ خود بازمی‌دارد، دربارهٔ هوش مصنوعی هم صادق است. وقتی تصمیم می‌گیرید کار چگونه تقسیم و واگذار شود، چهار پرسش بپرسید:

1. **اندازه** — کار برای یک نشست چند توکن است؟
2. **استقلال** — آیا می‌توان آن را بدون از دست دادنِ اثربخشی تقسیم کرد؟

3. **تفکیک وظایف** — آیا انجامِ هر دو توسط یک عامل تضاد منافع می‌سازد؟
4. **بازبینی‌پذیری** — اگر بازبینی بسیار ارزان‌تر از انجام است، از چند عاملِ بازبین استفاده کنید.

پیش از انتشار — بررسیِ ۹۰ ثانیه‌ای

> 1. آیا مسئلهٔ واقعی را تعریف کردم، یا فقط نخستین مسئله‌ای را که به ذهنم رسید؟
> 2. آیا کار در یک بستهٔ زمینه لنگر انداخته است، یا در حافظهٔ مدل؟
> 3. آیا یک عاملِ مستقل آن را بازبینی کرد — نه همان عاملی که ساختش؟
> 4. آیا همهٔ آزمون‌ها به‌صورت محلی پاس می‌شوند، و آیا تست‌های نفوذِ مکرر را از سر گذرانده است؟
> 5. آیا انسانی سیستمِ کاملاً در حال اجرا را به کار برده است؟

اگر پاسخِ هر پرسش «بله» است، وضعیت شما به‌مراتب بهتر از بیشترِ افراد است.

این یک راهنمای کلی و فراگیر دربارهٔ فرایند ساخت است — در سطحِ اصول و ترتیب می‌ماند و نمی‌تواند جزئیاتِ خاصِ یک پروژهٔ منفرد را پوشش دهد. برای راهنماییِ مفصل و متناسب با موقعیت شما، تماس بگیرید.

دربارهٔ Rehm KI Consulting

ما به کسب‌وکارها کمک می‌کنیم درست با هوش مصنوعی بسازند — با همان انضباطِ مهندسی که یک سیستمِ قابل‌اعتماد را از یک دموی چشمگیر جدا می‌کند. پاسخ‌های صریح دربارهٔ اینکه هوش مصنوعی کجا جا دارد و کجا ندارد. بدون هیاهو.

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