Ocena

Jak sprawdzić, czy agent albo prompt naprawdę działa.

- Jak sprawdzić, czy to w ogóle działa

Jak sprawdzić, czy to w ogóle działa

> „Wygląda dobrze" nie jest miarą, bo wrażenie dobrej odpowiedzi i faktyczna poprawność rozjeżdżają się tym bardziej, im lepiej model potrafi brzmieć przekonująco. Ta strona pokazuje, jak zbudować minimalny, powtarzalny sposób sprawdzania, czy agent albo prompt faktycznie robi to, co ma robić, i jak wyłapać moment, w którym przestał.

Skróty użyte na tej stronie (2)

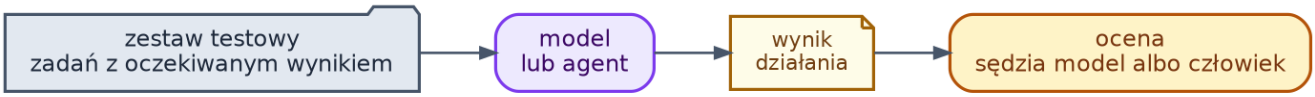
Skrót	Co znaczy
AI	Artificial Intelligence: sztuczna inteligencja, czyli programy naśladujące ludzkie rozumowanie
LLM	Large Language Model: duży model językowy, program przewidujący kolejne słowa tekstu

Czym jest ewaluacja

> **Pojęcie: ewaluacja.** Systematyczne porównanie odpowiedzi modelu albo agenta wobec z góry ustalonego, sprawdzalnego kryterium — nie subiektywne wrażenie osoby czytającej wynik, tylko powtarzalny test, który da tę samą, albo statystycznie tę samą, ocenę niezależnie od tego, kto go uruchamia i kiedy. Różnica między „przejrzałem dziesięć odpowiedzi i wyglądały nieźle" a ewaluacją jest taka sama jak różnica między code review na oko a zestawem testów jednostkowych — jedno łapie to, co rzuca się w oczy, drugie łapie regresję, której nikt by nie zauważył ręcznie.

Zestaw testowy zadań z oczekiwanym wynikiem

Punktem wyjścia każdej sensownej ewaluacji jest zestaw reprezentatywnych przypadków — kilkanaście do kilkudziesięciu zadań, z których każde ma zapisany oczekiwany wynik albo przynajmniej jasne kryterium, po czym poznać, że wynik jest poprawny. Dobry zestaw obejmuje nie tylko przypadki typowe, ale i krawędziowe — wejście niepełne, wejście sprzeczne, sytuację, w której poprawną odpowiedzią jest odmowa albo przyznanie się do niewiedzy, a nie wygenerowanie czegokolwiek za wszelką cenę. Zestaw testowy zbudowany wyłącznie z łatwych, oczywistych przypadków mierzy tylko to, że model radzi sobie z łatwymi, oczywistymi przypadkami — co samo w sobie rzadko jest tym, co chcemy wiedzieć.



Od zestawu testowego przez model do oceny wyniku (kliknięcie otwiera pełny rozmiar)

Test regresji po zmianie promptu albo modelu

> **Pojęcie: regresja.** Sytuacja, w której zmiana — nowa wersja promptu systemowego, upgrade modelu z jednej wersji na kolejną, zmiana opisu narzędzia — psuje coś, co wcześniej działało poprawnie, mimo że nikt tego nie planował ani nie zauważył od razu. Bez zestawu testowego uruchomionego przed zmianą i po niej nie ma jak stwierdzić regresji inaczej niż przez skargę kogoś, kto trafił na zepsuty przypadek już na produkcji. Z zestawem testowym różnica jest widoczna od razu: uruchamiamy ten sam zestaw zadań na starej i nowej wersji, porównujemy wyniki punkt po punkcie, i dopiero brak różnicy, albo różnica wyłącznie na plus, daje podstawę, żeby zmianę wdrożyć dalej.

Sędzia-model i jego ograniczenia

> **Pojęcie: sędzia-model (LLM-as-judge).** Technika, w której ocenę odpowiedzi jednego modelu powierza się drugiemu wywołaniu modelu, opisując mu kryteria oceny w formie rubryki — bo ręczne ocenianie setek przypadków przy każdej zmianie nie skaluje się, a dokładne dopasowanie tekstowe, czyli sprawdzenie, czy odpowiedź brzmi identycznie jak

> wzorcowa, zwykle nie ma sensu tam, gdzie ta sama poprawna treść da się wyrazić na wiele różnych sposobów.

Ograniczenia tej techniki są realne i warto je znać, zanim zaufa się jej bez zastrzeżeń. Sędzia-model bywa stronniczy na korzyść odpowiedzi wygenerowanych przez model z tej samej rodziny, co on sam. Ocena tego samego przypadku potrafi się różnić między uruchomieniami, z powodów opisanych na stronie Mapa pojęć: model, token, kontekst, prompt, temperatura — nawet niska temperatura nie daje pełnej powtarzalności. Sędzia-model radzi sobie gorzej przy zadaniach wymagających głębokiej wiedzy specjalistycznej albo subtelnej oceny niż przy zadaniach z jasnym, łatwym do zweryfikowania kryterium. Z tych powodów rubrykę sędziego warto od czasu do czasu skalibrować względem oceny człowieka na próbce przypadków — jeśli ocena sędziego systematycznie rozjeżdża się z oceną człowieka, problem leży w rubryce albo w samym sędzim, nie w testowanym modelu.

Metryki

Poprawność. Odsetek przypadków testowych spełniających ustalone kryterium — podstawowa miara, ale zawsze wobec konkretnego, spisanego kryterium, nigdy na oko.

Koszt. Liczba tokenów, a więc i pieniędzy, zużytych na jedno zadanie — istotna metryka sama w sobie, bo poprawa poprawności o kilka procent kosztem dziesięciokrotnie droższego wywołania nie zawsze jest opłacalną zamianą.

Czas. Opóźnienie odpowiedzi i liczba kroków pętli agenta potrzebnych do ukończenia zadania — rosnąca liczba kroków bywa wczesnym sygnałem, że agent zaczął błądzić, jeszcze zanim ostateczny wynik wyjdzie błędny.

Odsetek zadań dokończonych bez interwencji człowieka. Miara autonomii faktycznej, nie deklarowanej — ile zadań z zestawu testowego agent doprowadził do końca samodzielnie, bez potrzeby poprawki, dopytania czy ręcznej korekty w trakcie. Organizacja badawcza METR mierzy pokrewną wielkość na dużą skalę dla modeli frontowych — długość zadania, liczoną czasem, jaki zajęłoby ono człowiekowi, które model kończy poprawnie w około połowie prób — i pokazuje, że ta długość rosła wykładniczo, podwajając się z każdą kolejną generacją modeli w tempie liczonym w miesiącach. To ciekawa i konkretna liczba, ale warto traktować ją ostrożnie: mierzy głównie zadania programistyczne, tempo podwajania samo się zmieniało w ostatnich latach, a ekstrapolacja trendu w przyszłość jest z natury sporna — sygnalizujemy to wprost, zamiast udawać, że trend jest pewnikiem.

Jak wyłapać cichą degradację

Pojęcie: cicha degradacja. Pogorszenie jakości, które nie ujawnia się przez żaden komunikat błędu ani awarię — model po prostu zaczyna częściej mylić się w podobnych przypadkach, a nikt tego nie zauważa, dopóki nie napłynie skarga albo nie zbierze się na tyle błędów, żeby wzorzec stał się widoczny gołym okiem. Źródłem bywa cicha aktualizacja modelu po stronie dostawcy, pozornie niewinna zmiana w opisie narzędzia albo w prompcie systemowym, albo dryf w danych, na których agent teraz pracuje, w porównaniu do danych, na których go pierwotnie sprawdzaliśmy. Jedyna praktyczna obrona to uruchamianie tego samego zestawu testowego regularnie, niezależnie od tego, czy ktoś świadomie coś zmienił — coś w rodzaju ciągłej integracji, tylko dla promptów i konfiguracji agenta zamiast dla kodu — z alertem, gdy wynik spadnie poniżej ustalonego progu.

Logi i powtarzalność

Każde uruchomienie testu warto zapisać w całości: pełny kontekst, jaki widział model, dokładna wersja modelu i promptu, parametry takie jak temperatura, i sam wynik razem z oceną. Bez tego zapisu odtworzenie regresji po fakcie polega na zgadywaniu, co dokładnie się zmieniło — z zapisem staje się prostym porównaniem dwóch uruchomień, dokładnie tak jak porównuje się dwie wersje kodu przy pomocy różnicy w systemie kontroli wersji.

Fiszka

◫◫ PLIK KONFIGURACYJNY

Szkielet pojedynczego przypadku testowego

```
id: KrotkiIdentyfikatorPrzypadku
zadanie: OpisPolecenia
wejscie: DaneWejscioweLubOdwolanieDoPliku
oczekiwany_wynik: OpisTegoCoUznajemyZaPoprawne
kryterium: automatyczne | sedzia-model | reczne
wersja_modelu: NazwaIWersjaModeluWMomencieTestu
```

Pole `kryterium` decyduje o tym, jak dany przypadek jest sprawdzany — automatyczne dopasowanie działa tylko przy zadaniach o jednoznacznym, ścisłym wyniku (np. czy skrypt zwrócił kod wyjścia zero), sędzia-model przy zadaniach opisowych z wieloma poprawnymi sformułowaniami, a ręczne tam, gdzie żadne z powyższych nie daje wystarczającego zaufania — na przykład przy pierwszym uruchomieniu nowego zestawu, zanim zaufamy

automatycznej ocenie.

Przykład:

```
id: t014
zadanie: przygotuj procedurę dołączenia RHEL10 do domeny ad.zielona.example
wejscie: opis środowiska z r1-02, brak dodatkowych plików
oczekiwany_wynik: procedura zawiera realm join, nie wymaga edycji /etc/hosts,
  każdy krok ma komendę i jedno zdanie uzasadnienia
kryterium: sedzia-model
wersja_modelu: claude-sonnet-5, effort=high
```

Z życia wzięte

□□ Z ŻYCIA WZIĘTE

Zespół Oli od miesiąca używa tego samego agenta do przeglądu pull requestów w Gitea. Po cichej aktualizacji modelu po stronie dostawcy nikt nic nie zmienia w konfiguracji, a mimo to trzy dni z rzędu agent przepuszcza pull requesty z oczywistymi błędami literowymi w komunikatach commitów. Zespół uruchamia swój zestaw dwudziestu testowych przypadków co noc, więc spadek wyniku widać już następnego ranka, zanim ktokolwiek zdążył się poskarżyć. Ola wraca tymczasowo do poprzedniej wersji promptu systemowego, dopóki nie sprawdzi, co dokładnie się zmieniło.

□□ Ćwiczenia

Zadania sprawdzające.

1. Po cichej aktualizacji modelu przez dostawcę zespół zauważa więcej skarg na jakość odpowiedzi agenta, ale żaden pojedynczy przypadek nie wygląda na oczywisty błąd. Co powinno było wykryć ten problem, zanim dotarł do skarg?

Odpowiedź

Regularne, automatyczne uruchamianie tego samego zestawu testowego niezależnie od tego, czy ktoś świadomie zmienił prompt czy konfigurację — cicha aktualizacja modelu jest dokładnie

tym rodzajem zmiany, której nikt nie zainicjował świadomie po naszej stronie, więc jedyny sposób, żeby ją złapać, to porównanie wyniku testów sprzed i po, nie czekanie na skargi.

2. Sędzia-model konsekwentnie ocenia wyżej odpowiedzi wygenerowane przez model z tej samej rodziny co on sam, nawet gdy odpowiedzi innego modelu są równie dobre pod względem merytorycznym. Jak to sprawdzić i co zrobić, jeśli się potwierdzi?

Odpowiedź

Sprawdzamy to, zestawiając oceny sędziego z oceną człowieka na próbce identycznych przypadków — jeśli rozjazd jest systematyczny i w jedną stronę, mamy potwierdzoną stronniczość. Rozwiązaniem bywa zmiana modelu pełniącego rolę sędziego na inny niż oceniane modele, albo dodanie do rubryki konkretnych, sprawdzalnych kryteriów zamiast ogólnego „która odpowiedź jest lepsza", co ogranicza pole do subiektywnej preferencji.

Źródła: LLM Agent Evaluation Metrics, Confident AI: https://www.confident-ai.com/blog/llm-agent-evaluation-complete-guide — LLM-as-a-Judge w 2026, DeepEval: https://deepeval.com/blog/llm-as-a-judge — A new Moore's Law for AI agents, METR / AI Digest, metryka horyzontu czasowego: https://theaidigest.org/time-horizons

Uwaga: liczby o tempie podwajania horyzontu czasowego pochodzą z ekstrapolacji ograniczonego zestawu, głównie zadań programistycznych — traktujemy je jako ilustrację trendu, nie jako pewnik przenoszący się wprost na nasze własne zadania.