Kontrolowane wykonywanie narzędzi

Rejestr, walidacja argumentów i jedna próba na turę

Czas	Pracujesz z	Rezultat
około 45 min	Testy bez LLM	tool_tests.jsonl i tool_trace.jsonl

Sprawdzisz trzynaście scenariuszy wykonawcy. RAG zastępuje kontrolowany dubler testowy, aby można było policzyć wywołania i sprawdzić odrzucenia bez kosztu modelu.

Cel i rezultat rozdziału

Udowodnisz, że niepoprawne wywołanie nie uruchamia narzędzia, a każda próba zostawia uporządkowany wynik.

Po zakończeniu masz

- macierz trzynastu testów i obserwowane statusy;
- ślad również dla wczesnego odrzucenia;
- dowód limitu jednej próby i kontroli pochodzenia źródeł.

Zanim zaczniesz

Rozpakuj cały pakiet Student. Zachowaj razem notebooki, dane, kod, checkpointy i wyniki. Ścieżki w karcie są liczone od Student. Przygotowanie stanowiska opisuje instrukcja pracy z modułem 5.

Otwórz notebook

```
notebooki/M12_M5_R5_kontrola_narzedzi_start.ipynb
```

Materiał wejściowy	Do czego służy
kod/m5_tools.py	TOOL_REGISTRY, TOOL_VALIDATORS oraz execute_tool_call().
kod/m5_exercises.py	run_r5(): wejścia, dubler RAG i liczniki wykonania.

Testy bez LLM. Czas karty dotyczy ćwiczenia, bez instalacji i oglądania prezentacji. Materiał: M12-M5-v5, slajdy 191–241.

Praca krok po kroku

Krok 1 Przygotuj testy — Uruchom Przygotowanie w notebooku R5.

- Backend wczytuje lokalny indeks i checkpoint; żaden model nie jest wywoływany.
- Testowy RAG zwraca oznaczony wynik dublera, a nie odpowiedź rzeczywistego modelu.

Krok 2 Prześledź granicę wykonania — Otwórz execute_tool_call() oraz walidatory trzech narzędzi.

- Sprawdź kolejność: limit próby, nazwa, argumenty i stan, wywołanie, wynik.
- Znajdź odrzucenie bool w top_k i similarity_threshold oraz obcych pól.

Krok 3 Uruchom macierz — Wykonaj Próby i przeczytaj trzynaście rekordów.

- Porównaj expected_status, observed_status, rag_stub_calls i passed.
- Źródła i metadane są odczytywane lokalnie; run_shell nie należy do rejestru.

Krok 4 Sprawdź odrzucenia — Znajdź run_shell, ścieżkę zamiast document_id, bool i pole command.

- Dla odrzuconych prób licznik testowego RAG musi pozostać zerowy.
- W teście drugiej próby pierwsze wywołanie działa, drugie jest blokowane; licznik wynosi jeden.

Krok 5 Sprawdź błędy i ślad — Odczytaj przypadki wyjątku, zbyt dużego wyniku i brakujących źródeł.

- Każde wykonanie lub odrzucenie ma wpis w tool_trace.jsonl.
- Dwie próby testu limitu należą do tej samej tury, więc współdzielą jej result_id.

Krok 6 Zapisz uzasadnienia — Uzupełnij trzy pola oceny dla każdego wyniku i zapisz notebook.

- passed sprawdza kontrakt techniczny, a nie jakość odpowiedzi RAG.
- Opcjonalnie zmień jeden warunek w kopii walidatora i pokaż, który test wykrywa regresję.

Ważne: ponowne Przygotowanie tworzy nowy katalog. Nowe wykonanie oceń osobno. Po zmianie komentarzy uruchom wyłącznie ocenę i zapis; zachowaj surowe wyniki.

Punkty kontrolne pracy

Po etapie	Masz sprawdzić	Gdy wynik nie pasuje
Po macierzy	Wszystkie trzynaście passed ma wartość true.	Porównaj obserwowany status z kontraktem i kodem.
Po odrzuceniu	Nie wykonano testowego RAG.	Sprawdź rag_stub_calls oraz wczesny return.
Po śladzie	Każda próba ma rekord z ID rozmowy i wyniku.	Dla testu limitu oczekuj dwóch wpisów jednej tury.

Nazewnictwo i zapis wyników

Wyniki zapisują się w wyniki/R5/<data_id>/. Odczytaj rzeczywisty katalog z komunikatu Wyniki. Główny rezultat: tool_tests.jsonl i tool_trace.jsonl.

Zapisz notebook przez Ctrl+S. Oceny zapisują się jako ocena_<id>.json; szablon ocena_do_uzupelnienia.json pozostaje bez zmian. Sposób oddania określa prowadzący.

Lista kontroli przed zakończeniem

- □ Sprawdziłem nieznaną funkcję i niepoprawne argumenty.
- □ Potwierdziłem odrzucenie bool i dodatkowego pola.
- □ Brak źródeł daje kontrolowane doprecyzowanie.
- □ Wykryłem wyjątek i przekroczenie rozmiaru wyniku.
- □ Druga próba nie uruchamia narzędzia.
- □ Oddzielam test dublera od rzeczywistego RAG.

Gdy pojawi się problem

Objaw	Najpierw sprawdź
Test ma passed=false	Sprawdź status, kod błędu i licznik wywołań.
bool przeszedł walidację	Użyj type(value), ponieważ bool dziedziczy po int.
Brak śladu dla wczesnego return	Zapisuj wynik w kodzie wywołującym wykonawcę.
Powtórzony result_id w śladzie	Dwie próby mogą należeć do jednej tury; wyniki tur mają własne unikatowe ID.

Zasada końcowa: zachowaj oryginalny wynik i własne uzasadnienie. Błędna odpowiedź lub decyzja modelu jest materiałem do analizy, nie powodem do poprawiania surowego zapisu.