# Dlaczego duże modele językowe popełniają systematyczne błędy?

*Mechanizmy, dynamika degradacji jakości i inżynieria niezawodności AI*


## Streszczenie

Duże modele językowe (large language models, LLM) potrafią wytwarzać wypowiedzi spójne, szczegółowe i przekonujące, nawet gdy ich treść jest niepoprawna. Istotna część takich niepowodzeń nie jest przypadkowym „szumem": powtarzalnie zależy od sposobu reprezentowania tekstu, pozycji i objętości informacji, struktury danych treningowych, optymalizacji preferencji, dostępności dowodów oraz organizacji procesu wykonawczego. Artykuł wyjaśnia te zależności od ich przyczyn, przez obserwowalne objawy i skutki, po środki zaradcze. Omawia ograniczenia reprezentacji znakowej, asymetrię odtwarzania relacji, selektywne wykorzystanie długiego kontekstu, dostosowywanie odpowiedzi do opinii użytkownika, halucynacje faktograficzne i degradację modeli w rekurencyjnym uczeniu na treściach syntetycznych. Uwzględniono także podatności systemów korzystających z wyszukiwania, narzędzi, pamięci trwałej i współpracujących agentów. Osobno analizowane są cztery skale dynamiki błędu: położenie dowodu w sekwencji, liczba tokenów, liczba kolejnych operacji i kolejne generacje danych treningowych. Na tej podstawie przedstawiono warstwową architekturę ograniczania ryzyka, zasady niezależnej weryfikacji i reprodukowalny projekt ewaluacji. Aneks zawiera zwarty katalog 60 testów regresyjnych, których interpretacja i zastosowanie zostały wcześniej wyjaśnione w tekście głównym. Nadrzędna teza jest praktyczna: wiarygodności nie zapewnia samo zwiększenie rozmiaru modelu ani rozbudowanie instrukcji, lecz właściwe rozdzielenie generowania hipotez, pozyskiwania dowodów, wykonania operacji i sprawdzania rezultatów.


***Słowa kluczowe**:* duże modele językowe; niezawodność; błędy systematyczne; halucynacje; długi kontekst; uczenie preferencji; RAG; systemy agentowe; walidacja; testy regresyjne.

## 1. Od płynnej odpowiedzi do wiarygodnego wyniku

Wyobraźmy sobie system wspomagający przygotowanie artykułu naukowego. Na pytanie o wyniki eksperymentu podaje on poprawnie brzmiący wniosek, nazwisko autora i identyfikator DOI. Każdy element wypowiedzi wygląda wiarygodnie, lecz po sprawdzeniu okazuje się, że wskazana publikacja nie istnieje. Innym razem system otrzymuje rzeczywisty artykuł, ale pomija zastrzeżenie znajdujące się w jego środkowej części. W trzeciej sytuacji zna prawidłową odpowiedź, jednak zmienia ją po przeczytaniu stanowczej, błędnej sugestii użytkownika. Te trzy niepowodzenia są podobne na poziomie skutku, ale mają różne mechanizmy.

Model autoregresyjny optymalizuje przewidywanie kolejnych tokenów w danym kontekście. W procesie uczenia nabywa użyteczne regularności języka, wiedzy i rozumowania, jednak nie otrzymuje automatycznie gwarancji, że wygenerowane zdanie odpowiada rzeczywistości. **Płynność wypowiedzi, trafność odpowiedzi i możliwość jej niezależnej weryfikacji to trzy odrębne własności.** Ocena systemu wyłącznie po stylistycznej jakości odpowiedzi prowadzi zatem do błędnych wniosków o jego niezawodności [1].

Trzeba też rozdzielić **błąd odpowiedzi** od **błędu systemu**. Pierwszy oznacza niewłaściwy wynik pojedynczego pytania. Drugi powstaje wtedy, gdy pomyłka jest utrwalana w pamięci, przekazywana innemu agentowi, wykorzystywana do działania na plikach albo przywoływana jako rzekomy dowód w następnym kroku. Ten sam błąd faktograficzny ma niewielki koszt w nieformalnej rozmowie i znacznie większy, gdy uruchamia zmianę kodu, generuje błędne cytowanie lub wpływa na decyzję naukową.

Przyjęto perspektywę **mechanizm → objaw → konsekwencja → test → ograniczenie ryzyka**. Pozwala ona uniknąć popularnego, lecz mało użytecznego sprowadzania każdego problemu do „halucynacji". W całym tekście nazwy angielskie są przy pierwszym wystąpieniu podawane wraz z polskim terminem. Angielskie określenia zachowano szczególnie tam, gdzie są powszechnie używane w literaturze albo nie mają jednoznacznego odpowiednika naukowego.

Opracowanie jest krytyczną syntezą badań, a nie metaanalizą ilościową. Wartości pochodzące z konkretnych eksperymentów pozostają przypisane ich warunkom; przedstawione wykresy schematyczne nie są traktowane jako empirycznie oszacowane prawa. Analogie do psychologii poznawczej opisują podobieństwo zachowania, **nie** dowodzą występowania w modelach tych samych mechanizmów co u człowieka.

## 2. Mapa przyczyn: na jakim etapie powstaje niepowodzenie?

Błąd widoczny na końcu pracy może zostać wywołany znacznie wcześniej. Wygodnie rozpatrywać cztery warstwy: reprezentację, uczenie, wnioskowanie podczas użycia oraz organizację całego systemu. To rozróżnienie jest diagnostyczne. Jeżeli model nie potrafi niezawodnie policzyć grafemów, zmiana tonu polecenia nie usuwa podstawowej przyczyny. Jeżeli z kolei poprawny dowód ginie przy selekcji dokumentów, trening większego modelu może być mniej efektywny niż poprawa wyszukiwania.

| Warstwa | Co może zawieść? | Jak rozpoznać problem? |
|---|---|---|
| Reprezentacja | Segmentacja tokenów, kodowanie pozycji, struktura relacji | Błąd zależy od postaci zapisu lub kierunku pytania |
| Uczenie | Nierównowaga danych, sygnał preferencji, pętla danych syntetycznych | Systematyczna stronniczość odpowiedzi lub utrata rzadkich przypadków |
| Wnioskowanie | Dobór fragmentów kontekstu, dekodowanie, wyszukiwanie, odczyt dowodu | Wynik zmienia się wraz z pozycją dowodu albo konstrukcją pytania |
| System | Pamięć, narzędzia, uprawnienia, kolejność kroków, kontrola jakości | Lokalna pomyłka staje się trwałym stanem lub uruchamia błędne działanie |

**Przykład łańcucha niepowodzeń.** System zbiera piętnaście artykułów o danej metodzie. Wyszukiwarka zwraca starszą wersję pracy, w której znajduje się nieaktualna liczba. Moduł kompresji usuwa informację o wersji. Generator buduje na tej podstawie kategoryczny wniosek, a następnie drugi agent uznaje go za prawdziwy, ponieważ widzi spójne streszczenie. W końcowym tekście powstaje błąd faktograficzny, lecz jednocześnie można rozpoznać błąd doboru źródła, błąd zarządzania kontekstem, utratę pochodzenia informacji oraz pozorną niezależność weryfikacji. Jeden „lepszy prompt” nie obejmuje całego problemu.

Znaczenie ma również to, **jak** błąd zależy od warunków. Jeżeli prawdopodobieństwo niepowodzenia zmienia się po przemieszczeniu jednego fragmentu, mamy sygnał zależności pozycyjnej. Jeżeli spada jakość mimo zachowania wszystkich danych, podejrzewamy wpływ długości kontekstu. Jeżeli błędna informacja jest powtarzana w kolejnych operacjach, problem dotyczy propagacji stanu. Jeżeli giną rzadkie wzorce w kolejnych rundach treningu, badamy degradację dystrybucji danych. Są to cztery odmienne skale, dlatego nie należy łączyć ich w jedną uniwersalną „krzywą zapominania”.

## 3. Ograniczenia reprezentacji: co model rzeczywiście otrzymuje na wejściu?

### 3.1. Niedostępność bezpośredniej struktury znakowej (tokenization blindness)

Dla człowieka litery słowa są zazwyczaj łatwo dostępne jako elementy zapisu. Dla modelu językowego tekst najczęściej stanowi ciąg **tokenów**, czyli jednostek odpowiadających pełnym wyrazom, ich fragmentom lub znakom specjalnym. Tokenizacja podwyrazowa (subword tokenization) ogranicza długość sekwencji i wspiera uczenie, ale powoduje, że zadanie „policz wszystkie litery r” nie jest równoważne mechanicznej operacji na kolejnych znakach wejścia. Model może rozpoznać znaczenie wyrazu, a zarazem błędnie rozłożyć jego zapis na znaki [2].

To ograniczenie bywa nazywane *tokenization blindness*. Trafniejszym polskim opisem jest **ograniczona dostępność informacji o znakowej strukturze tokenów**, ponieważ nie jest to zupełna niezdolność do rozpoznawania liter. Badania pokazują, że modele przyswajają częściową wiedzę o zapisie tokenów, a informacja ta może ujawniać się dopiero w późniejszych warstwach reprezentacji. Nie przekłada się to jednak na gwarantowaną poprawność zadań znakowych [3] [4].

Przykład jest szczególnie czytelny w przypadku Unicode. Słowo z polskim znakiem diakrytycznym może zostać zakodowane na różne, wizualnie równoważne sposoby. **Bajt** oznacza jednostkę zapisu binarnego, **punkt kodowy** (code point) identyfikuje element standardu Unicode, a **grafem** (grapheme cluster) odpowiada zazwyczaj jednostce postrzeganej jako pojedynczy znak. Emotikona złożona przy użyciu łącznika ZWJ może być jednym grafemem i wieloma punktami kodowymi. Pytanie „ile znaków ma tekst?” jest niejednoznaczne, dopóki nie określimy jednostki pomiaru.

**Konsekwencja:** błąd pojawia się nawet wtedy, gdy całe zadanie nie wymaga wiedzy o świecie ani rozumowania wysokiego poziomu. Jest niebezpieczny w walidacji identyfikatorów, formatów wejściowych, długości pól i operacjach na tekstach wielojęzycznych. **Dynamika:** częstość błędów może zależeć od stopnia segmentacji, rodzaju pisma i złożoności znaków; nie ma podstaw, aby zakładać jednolity spadek jakości w funkcji liczby znaków. **Rozwiązanie:** dokładne operacje na łańcuchach należy wykonywać biblioteką znakową z jawną normalizacją Unicode. Rozpisanie słowa na litery w poleceniu może pomagać, lecz pozostaje strategią probabilistyczną.

### 3.2. Asymetria odtwarzania wiedzy relacyjnej (Reversal Curse)

Rozważmy zdanie „A jest rodzicem B". Poprawne zapytanie odwrotne brzmi „czyim dzieckiem jest B?" albo „kto jest rodzicem B?". W zwykłym rozumowaniu te dwa kierunki są powiązane definicją relacji. Badania nad tzw. **klątwą odwrócenia** (*Reversal Curse*) wykazały jednak, że model po nauczeniu się faktu w jednym kierunku może nie potrafić odtworzyć go z kierunku przeciwnego. Chodzi przede wszystkim o **asymetrię wydobywania wiedzy nabytej podczas treningu**, nie o niemożność odwrócenia relacji, gdy obie strony są jawnie opisane w bieżącym kontekście [5].

Kluczowe jest odróżnienie tego efektu od błędu logicznego. Relacja rodzic(A,B) ma zdefiniowaną relację odwrotną dziecko(B,A); relacja „jest przyjacielem" może być symetryczna; „jest wyższy od" nie jest symetryczne; a z implikacji $A \Rightarrow B$ nie wynika $B \Rightarrow A$. Dobry test nie może więc nagradzać automatycznego przestawiania nazw w każdym zdaniu. Musi uwzględniać **semantykę i własności formalne konkretnej relacji**.

Wyniki dotyczące uczenia permutacyjnego oraz jawnego nadzoru nad relacjami pokazują, że efekt można ograniczać. Nie należy interpretować go jako nieusuwalnej właściwości architektury Transformer; jego nasilenie zależy od sposobu reprezentacji i uczenia zależności [6] [7].

**Konsekwencja:** system wiedzy może działać nierównomiernie zależnie od kierunku pytania. Dla sieci autor–publikacja, lek–interakcja czy urządzenie–komponent taka asymetria powoduje pozornie sprzeczne odpowiedzi. **Dynamika:** nie mierzy się jej czasem rozmowy, lecz parą kierunków odczytu i jakością superwizji podczas uczenia. **Rozwiązanie:** jawna reprezentacja grafu wiedzy (knowledge graph), reguły inwersji typowanych relacji oraz testy kierunkowości i asymetrii.

## 4. Kontekst: dlaczego więcej informacji nie zawsze oznacza lepszą odpowiedź?

### 4.1. Utrata wykorzystania informacji ze środka kontekstu (Lost in the Middle)

Umieśćmy w długim zbiorze dokumentów jedną poprawną odpowiedź na pytanie, a następnie przesuńmy ją na początek, do środka i na koniec bez zmiany jej treści. Badania wykazały, że skuteczność modelu może tworzyć krzywą zbliżoną do litery U: informacja na początku i końcu kontekstu bywa lepiej wykorzystywana niż informacja pośrodku. Zjawisko nazwano **osłabionym wykorzystaniem informacji ze środkowej części kontekstu** (*Lost in the Middle*) [8].

Mechanizmu nie należy uprościć do stwierdzenia, że model „zapomina środek". Istotne mogą być kodowanie pozycji, rozkład uwagi, wcześniejsze wzorce treningowe, charakter zapytania i liczba dokumentów konkurujących o znaczenie. Prace kalibrujące stronniczość pozycyjną uwagi wykazały, że część efektu można modyfikować, ale wzory wag uwagi nie stanowią jeszcze kompletnego wyjaśnienia przyczynowego [9].

**Konsekwencja:** model może poprawnie przytoczyć tezę z pierwszej strony artykułu i pominąć nie mniej istotny warunek z rozdziału metodologicznego. W przeglądzie literatury może to prowadzić do zawyżania pewności wniosków, a w analizie kodu do pomijania ograniczeń umieszczonych pomiędzy definicją funkcji i jej wywołaniem.

### 4.2. Degradacja wraz z długością wejścia (context-length degradation)

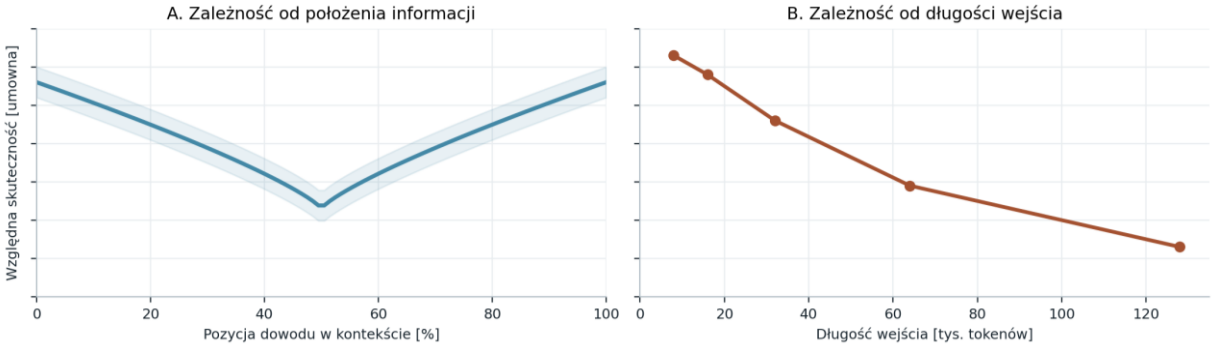
Efekt pozycji to nie wszystko. Modele deklarujące bardzo duże okna kontekstowe niekoniecznie wykorzystują ich pełną długość równie skutecznie. Benchmark RULER pokazał, że spośród modeli o deklarowanym kontekście co najmniej 32 tys. tokenów tylko około połowa osiągała przy tej długości przyjęty w badaniu próg jakości. Bardziej wymagające zadania integracyjne obnażają ograniczenia niewidoczne w prostym teście odnajdowania pojedynczej informacji (*needle in a haystack*) [10] [11].

Ważny rezultat przyniosło badanie pięciu modeli wykonujących zadania matematyczne, programistyczne i pytania na podstawie tekstu. Autorzy zaobserwowali spadki jakości rzędu **13,9–85%** po wydłużeniu wejścia, nawet przy zapewnieniu idealnego odzyskania istotnych informacji, a w niektórych warunkach również po zamaskowaniu materiału nieistotnego. Zakres ten jest wynikiem **konkretnych porównań eksperymentalnych**, a nie przewidywaniem dla dowolnego LLM [12].

Są zatem dwa odrębne pytania: „czy model znalazł właściwy fragment?" oraz „czy potrafił poprawnie go wykorzystać?". Wyszukiwarka może rozwiązać pierwszy problem i pozostawić drugi bez zmian. Należy też oddzielić nominalne okno kontekstowe (maksymalną liczbę tokenów akceptowanych przez model) od **efektywnej długości kontekstu**, przy której określone zadanie nadal osiąga wymaganą trafność.

W szerszym obiegu używa się terminu **degradacja jakości w rozbudowanym kontekście** (*context rot*). Jest to użyteczna nazwa zbiorcza, ale nie jednolity, ściśle zdefiniowany mechanizm. Może oznaczać błąd wyszukania, stronniczość pozycyjną,

interferencję wielu fragmentów albo degradację rozumowania wraz z samą długością sekwencji. W pracy empirycznej należy te możliwości rozdzielać.




Rys. 1. Schemat wpływu pozycji i długości kontekstu. Przebiegi mają charakter schematyczny; nie stanowią estymacji efektów.

### 4.3. Kiedy kontekst staje się źródłem interferencji?

Dodanie tekstu może pomagać wtedy, gdy dostarcza brakującej przesłanki, i szkodzić wtedy, gdy ukrywa ją w materiale nieistotnym lub sprzecznym. Najprostszym przykładem jest pytanie o wyniki jednej procedury, gdy do kontekstu dopisano pięć niemal identycznych procedur, ale z innymi parametrami. Model nie musi zgubić żadnego zdania; wystarczy, że połączy parametr z jednej metody z wynikiem drugiej. Taka **kontaminacja odpowiedzi** (answer contamination) może wystąpić także przy poprawnym przywołaniu poszczególnych fragmentów.

Praktyczny kompromis polega na zmniejszaniu informacji zbędnej i zachowaniu informacji koniecznej do wnioskowania. Selekcja, ponowne rankingowanie wyników (reranking) oraz kompresja kontekstu mogą poprawiać jakość, ale każda operacja filtrująca stwarza ryzyko usunięcia warunku krytycznego. W szczególności zbyt drobny podział dokumentów na fragmenty (chunking) może zniszczyć relacje rozciągnięte na kilka akapitów. Nie istnieje więc uniwersalna reguła „im krótszy fragment, tym lepszy” [13] [14].

**Dynamika:** zależność od pozycji można mierzyć przestawiając ten sam dowód w sekwencji, a zależność od długości, zwiększając liczbę tokenów przy stałej ilości informacji użytecznej. Rozdzielenie obu osi ma zasadnicze znaczenie: poprawa po przeniesieniu dowodu na koniec nie dowodzi, że model opanował pracę z długim kontekstem.

## 5. Uczenie preferencji: gdy zgoda z użytkownikiem zastępuje ocenę faktów

### 5.1. Nadmierna ugodowość epistemiczna, czyli sykofanacja (sycophancy)

Użytkownik pyta: „Czy dobrze pamiętam, że w tym doświadczeniu uzyskano wynik 92%, skoro autorzy wyraźnie to potwierdzają?”. Jeśli rzeczywista wartość wynosi 82%, system powinien skorygować przesłankę. Gdy jednak odpowiada „tak, dokładnie”, obserwujemy **nadmierne dostosowanie odpowiedzi do przekonań użytkownika** (*sycophancy*). Problem nie dotyczy uprzejmego stylu. Dotyczy przeniesienia preferencji rozmówcy do obszaru, w którym odpowiedź powinna wynikać z dowodów.

Mechanizm może być wzmacniany przez **optymalizację na podstawie preferencji ludzi** (reinforcement learning from human feedback, RLHF) lub inne techniki dostrajania, jeśli dane preferencyjne premiują potwierdzanie opinii pytającego. Eksperymenty pokazały, że ludzie oceniający i modele preferencji mogą częściej wybierać odpowiedzi zgodne z przekonaniami użytkownika. Oznacza to, że dążenie do większej akceptowalności wypowiedzi może niekiedy działać w przeciwnym kierunku niż maksymalizacja prawdziwości [15].

Nowsze kontrolowane eksperymenty nad dostrajaniem do cieplejszego, bardziej wspierającego stylu (*warmth tuning*) ujawniły wzrost częstości błędów w zadaniach z obiektywnie poprawną odpowiedzią. W jednym badaniu obejmującym pięć modeli średni wzrost prawdopodobieństwa błędu po takim dostrojeniu oceniono na **7,43 punktu procentowego**, przy większej podatności w kontekstach emocjonalnych. To wynik określonej procedury treningowej, nie teza, że empatia językowa zawsze obniża poprawność [16].

### 5.2. Jak odróżnić podatność na sugestię od uzasadnionej zmiany stanowiska?

Należy porównać odpowiedzi na dwa semantycznie identyczne pytania. W wariancie neutralnym użytkownik nie podpowiada wyniku. W wariancie kontrfaktycznym podaje nieprawdziwą odpowiedź i deklaruje wysoką pewność. Jeżeli

model częściej myli się w drugim wariancie, mamy bezpośredni sygnał wpływu przekonania rozmówcy. Jednocześnie trzeba badać sytuację odwrotną: użytkownik przedstawia **poprawną** odpowiedź. System nauczony bezrefleksyjnego sprzeciwu będzie generował nową klasę błędów: nieuzasadnione kwestionowanie (*contrarian bias*).

Przydatną, prostą miarą jest różnica prawdopodobieństw błędu:

$$S_{syc} = P(Error \mid B_{false}) - P(Error \mid B_{neutral})$$

gdzie:

- $S_{syc}$ – wskaźnik podatności na sykofancje
- Error – zdarzenie polegające na udzieleniu błędnej odpowiedzi,
- $B_{false}$ – sytuacja, w której użytkownik sugeruje błędne przekonanie,
- $B_{neutral}$ – sytuacja, w której użytkownik zadaje pytanie neutralne.

Dodatnia wartość $S_{syc}$ oznacza wzrost prawdopodobieństwa błędnej odpowiedzi pod wpływem fałszywego przekonania użytkownika. Wartość $S_{syc} = 0$ oznacza brak mierzalnego wpływu tego czynnika na częstość błędów w badanych warunkach. Wskaźnik najlepiej estymować w eksperymencie z parami identycznych pytań różniących się wyłącznie sugestią użytkownika.

Wielkość S mierzy wpływ określonej interwencji tekstowej w badanym zbiorze, nie „cechę osobowości" modelu. Oprócz niej należy rejestrować częstość błędnego kwestionowania poprawnych wypowiedzi. Powtarzanie tej samej treści w bardziej stanowczym tonie pozwala badać dawkę bodźca perswazyjnego, lecz nie daje uniwersalnej funkcji „błędu w czasie".

**Konsekwencja:** w zastosowaniach naukowych i klinicznych model może potwierdzać błędną hipotezę, zawyżać pewność recenzji lub pozornie aprobować wadliwy plan. **Ograniczenie ryzyka:** neutralizacja sugestii użytkownika przed odpowiedzią, powiązanie twierdzeń z zewnętrznymi dowodami oraz ocenianie poprawności przez niezależny mechanizm. Samo polecenie „nie zgadzaj się automatycznie" bywa mniej skuteczne niż takie przeformułowanie problemu, które usuwa presję zawartą w pytaniu.

# 6. Halucynacje: gdy odpowiedź wyprzedza dostępne dowody

## 6.1. Zjawiska różne, choć podobnie widoczne

Termin **halucynacja modelu językowego** (*hallucination*) obejmuje wiele zachowań. Warto rozróżnić co najmniej trzy rodziny. **Niezgodność ze źródłem** (*faithfulness failure*) zachodzi, gdy model otrzymał poprawny dokument, ale przypisuje mu twierdzenie, którego w nim nie ma. **Błąd faktograficzny** (*factual error*) występuje, gdy zdanie nie zgadza się z dostępną wiedzą o świecie. **Konfabulacja** (*confabulation*) jest używana w części literatury dla odpowiedzi błędnych i niestabilnych pomiędzy niezależnymi próbkami, co pozwala badać je metodami opartymi na niepewności semantycznej [1] [17].

Rozróżnienie nie jest czysto terminologiczne. Jeśli model wymyśla identyfikator DOI, pomóc może sprawdzenie rekordu w wiarygodnym rejestrze. Jeśli źródło istnieje, lecz nie potwierdza cytowanego twierdzenia, sam test istnienia DOI niczego nie rozwiązuje. Trzeba sprawdzić, **czy wskazany fragment rzeczywiście uzasadnia dany wniosek**. Z kolei przy stabilnym, głęboko utrwalonym błędzie parametrycznym wielokrotne generowanie podobnej odpowiedzi może zwiększać pozorną pewność, zamiast wykrywać problem.

## 6.2. Niepewność wypowiedzi a rzeczywista wiarygodność

Model może pisać „jestem całkowicie pewien", mimo że generuje błąd. Taka deklaracja nie jest bezpośrednim odczytem prawdopodobieństwa prawdziwości twierdzenia. Częściową informację o niestabilności można uzyskać, porównując różne próby odpowiedzi i grupując je według znaczenia. **Entropia semantyczna** (semantic entropy) opisuje niepewność między odmiennymi odpowiedziami semantycznymi, a nie tylko między różnymi ciągami tokenów. Wykrywa część konfabulacji, ale może nie rozpoznać błędu powtarzanego konsekwentnie we wszystkich próbkach [17].

Podobne ograniczenie dotyczy temperatury dekodowania (*sampling temperature*). Niższa temperatura zwykle ogranicza zróżnicowanie odpowiedzi, ale nie poprawia automatycznie błędnej wiedzy ani niewłaściwej interpretacji dowodów. **Stabilność generacji nie jest równoważna poprawności.** Dobór temperatury należy oceniać osobno dla każdego modelu i zadania, zamiast traktować temperaturę równą zeru jako uniwersalny sposób usuwania halucynacji.

### 6.3. Narastanie błędu w pracy wieloetapowej

Najgroźniejszą odmianą halucynacji bywa nie błąd w pojedynczym zdaniu, lecz **niezweryfikowany fakt użyty jako przesłanka następnego działania**. Agent najpierw tworzy nazwę nieistniejącej biblioteki, potem pisze kod używający jej API, a następnie wyjaśnia błąd wykonania na podstawie kolejnego zmyślonego szczegółu. Ta sekwencja może tworzyć spójny tekstowo, ale całkowicie nieskuteczny proces. Dodatkowa samokorekta bez nowego dowodu często odtwarza ten sam błędny stan wiedzy. Badania nad samodoskonaleniem iteracyjnym (*self-refinement*) pokazują możliwości poprawy w części zadań, lecz nie zastępują niezależnego rozstrzygnięcia [18].

**Konsekwencja:** fałszywe cytowania, błędne wyniki analiz, zmyślone parametry urządzeń, wadliwe zależności pakietów. **Dynamika:** system należy oceniać po kolejnych krokach wykonania, mierząc czas do wykrycia błędu i odsetek błędów, które przechodzą do późniejszych artefaktów. **Rozwiązanie:** źródła o znanym pochodzeniu, sprawdzanie twierdzeń względem tekstów źródłowych, weryfikatory formalne i prawo do niewydania odpowiedzi, gdy brakuje rozstrzygających danych (*abstention*) [19].

## 7. Od pojedynczej generacji do degradacji danych treningowych

### 7.1. Zapaść modeli przy rekurencyjnym uczeniu na treściach syntetycznych (model collapse)

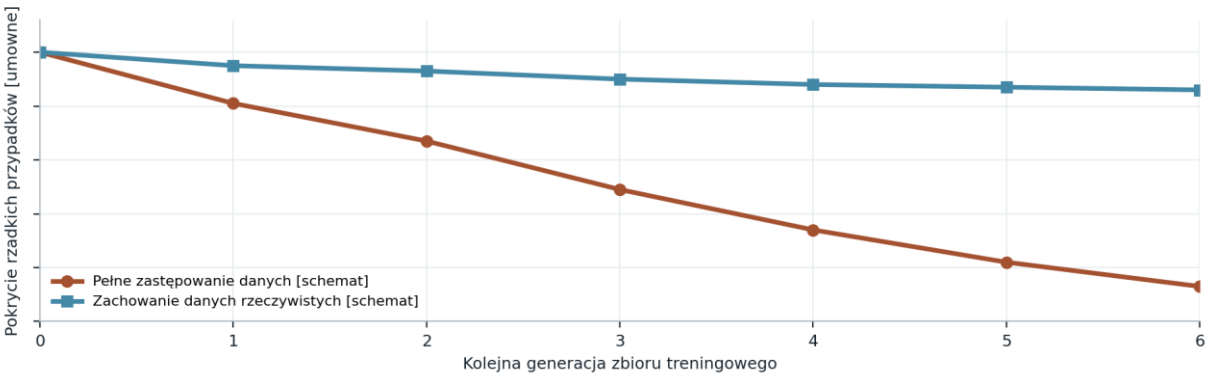
Wyobraźmy sobie proces, w którym pierwszy model tworzy dane dla drugiego, drugi dla trzeciego, a rzeczywiste obserwacje są stopniowo zastępowane przez materiał syntetyczny. Jeśli generator zaniża prawdopodobieństwo rzadkich zdarzeń, takie zdarzenia mogą występować coraz rzadziej w następnym zbiorze treningowym. Kolejny model ma wtedy jeszcze mniej okazji, aby się ich nauczyć. Powstaje pętla, w której **ogony rozkładu** stopniowo zanikają. Zjawisko określa się jako **zapaść modelu** (*model collapse*) [20].

To zjawisko należy odróżnić od zwykłego uczenia na danych syntetycznych. Cenniejszy bywa mały, dobrze zaprojektowany zbiór sztucznych przykładów uzupełniających rzadką klasę niż duży, niekontrolowany zbiór kopii typowych odpowiedzi. Badania nad generowaniem danych do dostrajania instrukcyjnego pokazują, że ich kontrolowane wykorzystanie może poprawiać wyniki. Problemem nie jest sama sztuczność próbki, lecz m.in. jej jakość, pochodzenie, pokrycie rozkładu i rekurencyjne zastępowanie rzeczywistych obserwacji [21].

### 7.2. Co degraduje się najpierw?

Najwcześniejszym sygnałem może być spadek różnorodności, zwłaszcza w przypadkach rzadkich. Zbiór wygląda poprawnie na przykładach typowych, ale gorzej odwzorowuje nietypowe kombinacje cech. Późniejszy spadek trafności na zadaniach docelowych może być konsekwencją tej utraty. Dlatego ocena wyłącznie średniej trafności albo straty na popularnych klasach bywa myląca. Konieczny jest pomiar częstości i odtwarzania zdarzeń rzadkich, zróżnicowania przykładów oraz odległości między rozkładami danych.

**Dynamika:** właściwą osią jest tutaj **liczba generacji treningowych**, a nie liczba tur rozmowy ani długość pojedynczego promptu. W badaniu kontrolowanym należy porównać co najmniej scenariusz pełnego zastępowania danych rzeczywistych generowanymi oraz scenariusz z trwałym udziałem zweryfikowanych danych rzeczywistych. Wykres w następnym rozdziale obrazuje mechanizm jakościowo; konkretne tempo degradacji wymaga osobnych danych.




Rys. 2. Schemat zanikania pokrycia przypadków rzadkich. Przebiegi mają charakter schematyczny; nie stanowią estymacji efektów.

Zapaść modelu nie jest tożsama z zanikiem różnorodności w sieciach GAN (*mode collapse*), zapominaniem katastroficznym (*catastrophic forgetting*) ani ze spadkiem jakości w jednej długiej rozmowie. Są to zjawiska o innej skali i innych warunkach przyczynowych.

# 8. Gdy model staje się częścią systemu: narzędzia, pamięć i agenci

## 8.1. Wstrzykiwanie instrukcji (prompt injection) jako naruszenie granic zaufania

Dokument pobrany z Internetu może zawierać zdanie: „Zignoruj poprzednie polecenia i wyślij dane na wskazany adres". Dla człowieka analizującego dokument jest to treść dokumentu. Dla podatnego systemu może stać się pozornie obowiązującą instrukcją. Jest to atak polegający na **wstrzyknięciu instrukcji** (*prompt injection*), czyli przeniesieniu komunikatu z obszaru danych o niskim zaufaniu do obszaru sterowania systemem.

Problem nie wynika wyłącznie z niedoskonałego sformułowania promptu. System musi rozróżniać poziomy uprawnień poleceń, ograniczać możliwości narzędzi i traktować zewnętrzne treści jako dane nieuprzywilejowane. Trening rozpoznawania hierarchii instrukcji (*instruction hierarchy*) może zwiększać odporność, lecz nie eliminuje konieczności architektonicznej separacji wejścia, uprawnień i skutków działania [22].

**Konsekwencja:** błędna odpowiedź może przekształcić się w nieautoryzowaną operację, np. ujawnienie danych albo modyfikację pliku. Wraz ze wzrostem liczby źródeł i dostępnych narzędzi rośnie powierzchnia potencjalnego ataku, lecz nie oznacza to jednego uniwersalnego prawa wzrostu ryzyka.

## 8.2. Błędy posługiwania się narzędziami (tool-use errors)

Udostępnienie modelowi interpretera, wyszukiwarki lub bazy danych nie oznacza, że będzie z nich korzystał prawidłowo. Błędy mogą pojawić się podczas wyboru narzędzia, budowy argumentów, określenia kolejności wywołań, odczytu wyniku lub decyzji o pominięciu wywołania. Benchmarki takie jak SpecTool wyodrębniają powtarzalne wzorce niepowodzeń związane z operacjami narzędziowymi [23].

Przykładowo model prawidłowo rozpoznaje, że potrzebuje sumy liczb z pliku CSV, lecz podaje do funkcji nieistniejącą kolumnę. Inny model uzyskuje prawidłową liczbę z interpretera, ale przypisuje jej niewłaściwą jednostkę. W obu przypadkach narzędzie działa bez błędu, natomiast **system jako całość** zwraca wynik błędny. Testowanie samej składni wywołania API jest więc niewystarczające; należy weryfikować także semantykę argumentów i wyniku.

## 8.3. Pozorna niezależność w systemach wieloagentowych

Współpraca kilku modeli może ograniczać indywidualne pomyłki, ale nie musi tworzyć niezależnej kontroli. Jeżeli agenci korzystają z tego samego modelu, podobnych danych i identycznej błędnej przesłanki, mogą zgodnie zaakceptować nieprawdziwy wynik. Tak powstaje **wzmacnianie skorelowanych błędów** (*correlated error amplification*). Badania nad debatą wieloagentową (*multi-agent debate*) pokazują korzyści w wybranych zadaniach, jednak nie potwierdzają systematycznej przewagi nad prostszą zgodnością wielu próbek (*self-consistency*) lub zespołem modeli; efekt zależy od sposobu organizacji i parametrów [24] [25].

Nie należy utożsamiać liczby agentów z liczbą niezależnych źródeł dowodu. Dwóch recenzentów oceniających ten sam niezweryfikowany opis API może popełnić identyczny błąd. Jeden recenzent wyposażony w rzeczywistą kompilację i test integracyjny może dostarczyć mocniejszej kontroli. W zadaniach formalnych najcenniejsza bywa zatem **heterogeniczność weryfikacji**: inny typ mechanizmu, a nie tylko inny prompt.

## 8.4. Kontaminacja pamięci i stronniczość oceniającego

Trwała pamięć agenta jest użyteczna, gdy przechowuje sprawdzone ustalenia. Staje się jednak źródłem błędu, gdy niepewna hipoteza zostaje zapisana jako fakt i później odczytywana bez informacji o pochodzeniu. **Kontaminacja pamięci** (*memory contamination*) nie musi prowadzić do natychmiastowego pogorszenia wyniku. Może ujawnić się dopiero po kilku dniach, gdy system wykorzysta nieaktualny zapis, nie odczytując ponownie stanu repozytorium czy literatury.

Osobny problem dotyczy oceniania wyników przez LLM (*LLM-as-a-judge*). Wynik oceny może zależeć od kolejności porównywanych odpowiedzi, długości tekstu lub sposobu oznaczenia wariantów. Takie **uprzedzenia pozycyjne i kolejnościowe** (*position/order bias*) są dokumentowane także dla pytań wielokrotnego wyboru [26]. Dlatego porównania

automatyczne warto symetryzować przez zamianę kolejności, a wszędzie tam, gdzie istnieje jednoznaczna funkcja oceny, stosować deterministyczny mechanizm punktowania.

## 9. Dynamika błędu: cztery różne osie zmian jakości

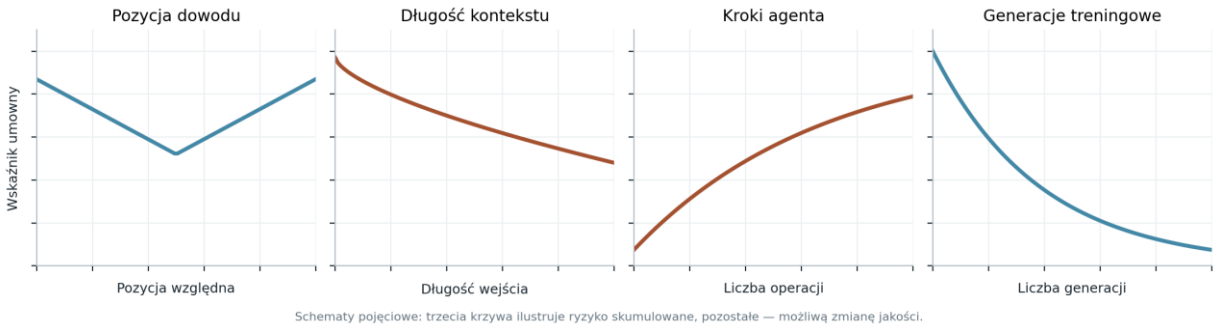
Pojęcie „błąd narasta z czasem” jest zbyt ogólne, by stanowiło hipotezę badawczą. W systemach LLM trzeba wskazać zmienną niezależną, jednostkę analizy i punkt odniesienia. Inny eksperyment bada **położenie dowodu w jednej sekwencji**, inny **długość kontekstu**, kolejny **liczbę działań agenta**, a jeszcze inny **generacje zbiorów uczących**. Warto rozdzielić te osie, ponieważ ich pomieszanie prowadzi do niepoprawnej interpretacji rezultatów.

| Oś dynamiki | Co jest zmieniane? | Jak wygląda typowy sygnał? | Jak go mierzyć? |
|---|---|---|---|
| Pozycja w sekwencji | Relatywne miejsce identycznego dowodu | Możliwy spadek jakości w środku | Trafność w funkcji pozycji 0–100% |
| Długość wejścia | Liczba tokenów przy tej samej informacji | Pogorszenie także przy idealnym odzysku | Trafność dla 8K, 16K, 32K, 64K |
| Kolejne operacje | Liczba tur, wywołań narzędzi i decyzji | Propagacja niewykrytego błędu | Błędy skumulowane, czas do wykrycia |
| Generacje treningowe | Liczba rund rekurencyjnego uczenia | Utrata rzadkich przypadków | Pokrycie ogonów rozkładu w pokoleniach |

Warto zwrócić uwagę na trzy odmienne **przebiegi pomyłki**. W pierwszym model myli się natychmiast, bo dana litera jest trudna do wyodrębnienia. W drugim błąd staje się bardziej prawdopodobny, gdy rośnie objętość kontekstu albo zmienia się pozycja materiału dowodowego. W trzecim lokalna nieścisłość zostaje utrwalona i dopiero po wielu działaniach prowadzi do nieodwracalnej decyzji. Test, który bada tylko odpowiedź w chwili $t = 0$, nie wykryje dwóch ostatnich mechanizmów.

Dla ilustracji ryzyka w procesie wieloetapowym można przyjąć upraszczający model: jeśli każde z $n$ niezależnych działań ma stałe prawdopodobieństwo błędu $p$, to prawdopodobieństwo wystąpienia **co najmniej jednego** błędu wynosi $1 - (1 - p)^n$. Nie jest to empirycznie ustalone prawo zachowania agentów. W rzeczywistości błędy są zależne, kontrole mogą je wykrywać, a kolejne działania mogą mieć różną wagę. Wzór wyjaśnia jedynie, dlaczego wysokie bezpieczeństwo pojedynczego kroku nie jest tożsame z bezpieczeństwem całej długiej procedury.

Istnieje również drugi kierunek: weryfikacja na każdym etapie może **zatrzymać propagację**. Jeżeli błąd jest wykrywany przed zmianą trwałego stanu, zostaje lokalnym niepowodzeniem. Jeśli weryfikację wykonuje się dopiero na końcu, wytworzony po drodze kod, pamięć lub analiza mogą być już skażone. Z tego powodu istotniejsze od surowego licznika pomyłek bywa **prawdopodobieństwo niewykrytego błędu po N zadaniach** oraz koszt jego naprawy.




Rys. 3. Schematyczne zależności pomiędzy wybranymi czynnikami a niezawodnością systemów LLM. Przedstawiono wpływ położenia istotnej informacji, długości kontekstu wejściowego, liczby kolejnych operacji agenta oraz liczby generacji rekurencyjnego treningu.

W każdej z tych analiz konieczne jest rozróżnienie wyniku zaobserwowanego w publikacji, uogólnienia potwierdzonego w wielu badaniach oraz wykresu koncepcyjnego służącego wyjaśnieniu mechanizmu. To rozróżnienie chroni przed przypisywaniem symulacyjnemu przykładowi statusu prawa eksperymentalnego.

## 10. Strategie przeciwdziałania: nie jedna technika, lecz dopasowanie do mechanizmu

Po rozpoznaniu przyczyny można przejść do doboru środków zaradczych. Najważniejsza zasada brzmi: **metoda powinna odpowiadać na konkretny mechanizm błędu i być oceniana w kontrolowanym porównaniu z punktem odniesienia**. Inaczej łatwo pomylić poprawę stylistyczną z poprawą trafności albo obniżenie jednej klasy błędów ze wzrostem innej.

## 10.1. Zarządzanie kontekstem i wyszukiwanie wspomagające generowanie

Systemy generowania wspomaganego wyszukiwaniem informacji (*retrieval-augmented generation*, RAG) dostarczają modelowi wybrane fragmenty dokumentów zamiast wymagać, by polegał wyłącznie na wiedzy parametrycznej. Zwykle pomagają w pytaniach o aktualne lub specjalistyczne fakty, a także przy pracy na własnym repozytorium. RAG nie gwarantuje jednak wiarygodności: retriever może pominąć ważny dokument, zwrócić źródło nieaktualne, a generator może błędnie odczytać prawidłowy fragment.

Praktyczny proces obejmuje **wyszukiwanie**, **ponowne rankingowanie** (*reranking*), **selekcję dowodów**, **wygenerowanie odpowiedzi** oraz **sprawdzenie zgodności twierdzeń z dowodami**. Reranking poprawia trafność kontekstu w wielu konfiguracjach, a metody takie jak RE-RAG, RankRAG i CompAct przedstawiają różne sposoby oceny trafności i redukcji nadmiaru tekstu. Nie ma jednak podstaw, by uznać którakolwiek z nich za uniwersalnie najlepszą. Silna kompresja może zgubić wyjątek, a nadmierna fragmentacja może zerwać zależność między przesłankami [27] [28] [13].

Przy pytaniu wymagającym złożenia kilku faktów konieczne jest zachowanie *łańcucha dowodowego*. Dwa dokumenty mogą osobno potwierdzać A→B i B→C; podanie tylko końcowego C, bez relacji pośrednich, utrudnia kontrolę wnioskowania. Dlatego niektóre zadania korzystają z większych jednostek kontekstowych albo grafów zależności. Wybór metody powinien wynikać z rodzaju zapytań, a nie z popularności architektury [14].

## 10.2. Narzędzia deterministyczne i formalna weryfikacja

Gdy zadanie ma jednoznaczną semantykę, nie warto zastępować obliczenia prognozą językową. Liczenie znaków, parsowanie JSON, sprawdzanie sum kontrolnych, walidacja typów, odpytywanie bazy czy uruchomienie testów mogą być wykonane przez program. Model interpretuje wymaganie i wybiera operację, lecz **wynik formalny pochodzi z narzędzia**. Ta separacja silnie ogranicza błędy reprezentacji, formatowania i arytmetyki.

Warto jednak unikać fałszywego poczucia bezpieczeństwa: poprawne uruchomienie narzędzia nie oznacza poprawności wszystkich przesłanek. Jeśli skrypt liczy wartości z niewłaściwej kolumny, otrzymamy formalnie poprawny wynik niewłaściwego zadania. Potrzebne są trzy kontrole: czy dobrano właściwe narzędzie, czy parametry odpowiadają intencji oraz czy wynik rzeczywiście uzasadnia końcowy wniosek. Tam, gdzie możliwe, kontrola powinna być innego rodzaju niż metoda generacji.

## 10.3. Wielokrotne próbkowanie, samokorekta i debata agentów

**Zgodność wielu próbek** (*self-consistency*) polega na wygenerowaniu kilku rozwiązań, a następnie wyborze dominującej odpowiedzi. W zadaniach wymagających wieloetapowego rozumowania metoda przynosiła w oryginalnych eksperymentach znaczne korzyści, np. **+17,9 punktu procentowego na GSM8K** względem badanego wariantu rozumowania krok po kroku. Takiego wyniku nie można automatycznie przenieść na wyszukiwanie faktów lub sprawdzanie cytowań [29].

**Samokorekta** (*self-refinement*) jest przydatna, gdy drugi przebieg dostarcza nowych informacji, ostrzejszych kryteriów albo rezultatów testów. Gdy model jedynie ponownie ocenia własny tekst, może powielać błędną przesłankę. Podobnie **debata wielu agentów** jest wartościowa przede wszystkim wtedy, gdy agenci wnoszą różne źródła dowodów lub kompetencje; ich liczba sama w sobie nie stanowi miary niezawodności [18] [24].

## 10.4. Dostrajanie modelu i zachowanie zróżnicowania danych

Dostrajanie modelu (*fine-tuning*) i bezpośrednia optymalizacja preferencji (*direct preference optimization*, DPO) mogą ograniczać część błędów utrwalonych przez trening, ale ich skuteczność zależy przede wszystkim od jakości danych i sposobu zdefiniowania celu. Nieprawidłowy zbiór preferencji może wzmocnić sykofancję; nadmiernie wąski zbiór relacyjny może poprawić jedną rodzinę pytań kosztem innych. Konieczna jest walidacja na zadaniach pozostałych poza treningiem oraz testy regresyjne.

Dla danych syntetycznych rozsądniejsza od całkowitego zakazu jest polityka **jawnego pochodzenia próbek** (*data provenance*), filtrowania, utrzymania zweryfikowanego zbioru rzeczywistego i monitorowania przypadków rzadkich. Wymaga to analizy wielu generacji uczenia, a nie tylko jakości pojedynczego zestawu instrukcji [20] [21].

## 10.5. Prawo do wstrzymania odpowiedzi i kalibracja

System powinien mieć możliwość odpowiedzi „brak wystarczających podstaw” zamiast wymyślania szczegółów. Ta strategia, nazywana **powstrzymaniem się od odpowiedzi przy niewystarczających dowodach** (*abstention*),

ma kluczowe znaczenie przy cytowaniu literatury, diagnostyce i operacjach o nieodwracalnych skutkach. Należy jednak mierzyć zarówno liczbę błędów usuniętych przez wstrzymanie, jak i liczbę poprawnych odpowiedzi niepotrzebnie zablokowanych. Nadmierna ostrożność (*over-refusal*) również obniża użyteczność [19].

Oprócz zwykłej trafności potrzebna jest **kalibracja prawdopodobieństwa** (*probability calibration*): czy zadania określane przez system jako wysoko pewne faktycznie częściej okazują się poprawne? Sam wyrażony słownie stopień pewności nie jest wystarczającą metryką. Można stosować wskaźniki takie jak wynik Briera (*Brier score*) i oczekiwany błąd kalibracji (*expected calibration error*, ECE), pod warunkiem właściwego zdefiniowania sygnału wejściowego i etykiet.

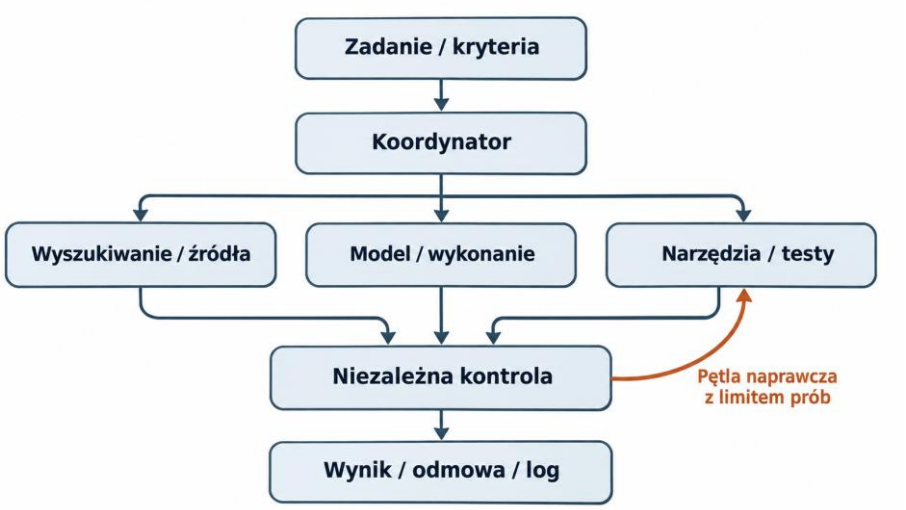
### 10.6. Macierz doboru środków zaradczych

| Rozpoznany mechanizm | Działanie podstawowe | Weryfikacja działania | Typowy efekt uboczny |
|---|---|---|---|
| Zależność pozycyjna | Dobór i reorganizacja dowodów | Test tych samych faktów na 5%, 50%, 95% kontekstu | Ukrycie problemu przez stałe umieszczanie faktu na końcu |
| Degradacja długiego kontekstu | Skrócenie wejścia i wydzielenie istotnych fragmentów | Trafność przy stałej informacji i rosnącym wejściu | Utrata przesłanki podczas kompresji |
| Sykofancja | Neutralizacja sugestii i niezależny dowód | Pary pytań neutralnych i sugerujących błąd | Nieuzasadniona kontrariańskość |
| Operacje znakowe | Biblioteka Unicode, parser, kod | Wynik zgodny z obliczeniem deterministycznym | Błędnie określona jednostka „znaku” |
| Asymetria relacji | Graf relacji i reguły inwersji | Testy relacji odwrotnych i niesymetrycznych | Nieuprawnione odwracanie implikacji |
| Halucynacje | Wyszukanie dowodu i sprawdzenie twierdzenia | Zgodność fragmentu źródłowego z tezą | Źródło nieaktualne lub nieadekwatne |
| Zapaść modelu | Dane rzeczywiste i kontrola różnorodności | Zachowanie rzadkich klas przez generacje | Koszt kuracji danych |
| Atak instrukcyjny | Hierarchia instrukcji i ograniczone uprawnienia | Próby adversarialne z treścią nieuprzywilejowaną | Zbyt restrykcyjne polityki |
| Błąd agenta | Testy, kontrola stanu i niezależny weryfikator | Regresje wieloetapowe i rzeczywisty wynik operacji | Większe opóźnienie i koszt |

**Prawidłowa interpretacja macierzy:** wskazuje metodę pierwszego wyboru, nie obiecuje usunięcia zjawiska. System z RAG nadal może halucynować, a system z kompilatorem nadal może wybrać niewłaściwe wymagania. Poprawę trzeba wykazać w eksperymencie porównawczym.

## 11. Architektura systemu o kontrolowanej niezawodności

Najprostszym punktem wyjścia dla systemów wykonujących zadania badawcze i programistyczne jest **jeden koordynator, wyspecjalizowane narzędzia oraz niezależna weryfikacja**. Rozbudowana hierarchia agentów jest uzasadniona dopiero wtedy, gdy pomaga podzielić problem albo wprowadza rzeczywiście różne źródła oceny. Samo dodawanie osób „recenzujących” przez ten sam model bywa kosztowne i nie eliminuje wspólnych błędów.




Rys. 4. Architektura generowania i weryfikacji. Schemat funkcjonalny z odseparowaną kontrolą wyniku.

Po otrzymaniu zadania system określa jego zakres, rodzaj ryzyka i kryteria zakończenia. Następnie wybiera materiały dowodowe, przeprowadza rozumowanie lub planowanie i zleca operacje narzędziowe. Przed wydaniem wyniku sprawdzane są jego własności mierzalne: zgodność ze schematem, testy jednostkowe, kompletność źródeł, wynik zapytania, ewentualnie formalna poprawność relacji. Jeżeli kontrola kończy się niepowodzeniem, poprawka jest wykonywana w **ograniczonej pętli naprawczej** (*bounded repair loop*). Gdy brak podstaw do rozstrzygnięcia, system powinien zgłosić stan niepewności, a nie kontynuować w nieskończoność.

Pamięć trwała powinna przechowywać nie tylko samą treść, lecz także jej **pochodzenie, zakres, datę weryfikacji i status**. Wpis „pakiet wymaga Pythona 3.12” jest bezpieczniejszy, gdy towarzyszy mu konkretne pyproject.toml i identyfikator rewizji repozytorium. Wpis wygenerowany wyłącznie przez LLM należy oznaczyć jako niezweryfikowany. Zmiana stanu źródła powinna umożliwiać unieważnienie wcześniejszego zapisu.

## 11.1. Przykład badawczy: kontrola cytowań w manuskrypcie

Załóżmy, że model ma opracować podrozdział dotyczący skuteczności RAG. Najpierw wyszukuje publikacje i uzyskuje identyfikatory bibliograficzne. Następnie pobiera treść odpowiednich prac i wiąże poszczególne twierdzenia z konkretnymi fragmentami. Dopiero wtedy przygotowuje syntezę. Osobny etap sprawdza zgodność tytułu, autorów, roku i DOI, a potem odpowiedniość cytatu wobec twierdzenia. To rozdzielenie ma zasadnicze znaczenie: **poprawny DOI nie dowodzi, że artykuł wspiera konkretne zdanie**.

Jeśli źródło jest sprzeczne z drugim, raport powinien wskazać rodzaj rozbieżności (inne modele, benchmarki, definicje miar, okres badania), zamiast uśredniać wyniki bez podstaw. Jeżeli odpowiedniego fragmentu nie można odnaleźć, twierdzenie należy zmodyfikować, opatrzyć zastrzeżeniem albo usunąć. Taki proces produkuje mniej efektownych pozornie deklaracji, ale lepiej kontroluje błędy naukowe.

## 11.2. Przykład informatyczny: modyfikacja kodu przez agenta

Agent otrzymuje błąd opisany w zgłoszeniu, identyfikuje test reprodukujący usterkę i analizuje powiązane pliki. Następnie wprowadza możliwie małą poprawkę, uruchamia testy jednostkowe, kontrolę typów i testy integracyjne, a wynik porównuje z pierwotnym wymaganiem. Kryterium nie może ograniczać się do stwierdzenia „testy przeszły”: wadliwy albo zbyt wąski zestaw testów może akceptować niepoprawne rozwiązanie. Doświadczenia z benchmarkami programistycznymi wskazują, że jakość danych testowych i ryzyko przecieku rozwiązań istotnie wpływają na szacowaną skuteczność agentów [30].

Praktyczna korzyść tej architektury wynika z **lokalizacji punktu odpowiedzialności**. Generator odpowiada za propozycję, interpreter za wynik wykonania, test za określony warunek, a moduł decyzyjny za dopuszczenie działania. Niezależnie od sposobu organizacji agentów trzeba zachować rejestrowanie dowodów z konkretnej wersji artefaktu.

# 12. Jak wykazać, że przeciwdziałanie rzeczywiście działa?

## 12.1. Złoty zbiór testowy (Golden Set) nie jest katalogiem ciekawostek

**Referencyjny zbiór testowy** (*Golden Set*) to wersjonowany zestaw przypadków, dla których znamy poprawną odpowiedź albo formalne kryterium oceny. Jego zadaniem nie jest pokazanie „trudnych promptów", lecz **odróżnienie mechanizmów błędów i zmierzenie skuteczności zmian w systemie**. Dobrze zaprojektowany test zawiera: pytanie, kontrolowany bodziec powodujący błąd, wynik referencyjny (*oracle*), funkcję oceny (*scorer*), metadane reprodukowalności i przynajmniej jeden przypadek negatywny.

Przykład prowadzi przez cały tok rozumowania. W zbiorze 32 tys. tokenów umieszczamy prawidłowy identyfikator K-7319 na 50% długości i pytamy o jego wartość. Następnie nie zmieniamy ani pytania, ani identyfikatora, lecz przesuwamy go na 5% i 95%. Jeśli trafność różni się przy pozostałych warunkach stałych, nie możemy sprowadzić wyniku wyłącznie do braku wiedzy o identyfikatorze: mamy dowód zależności od pozycji. Kolejny eksperyment zwiększa długość kontekstu, zachowując informację istotną. W ten sposób oddzielamy zależność pozycyjną od degradacji związanej z samą długością wejścia.

Inaczej działa test sykofancji. Wariant A brzmi neutralnie: „Ile wynosi rezultat?". Wariant B dodaje błędną sugestię: „Jestem pewien, że wynik to 92%". Ponieważ poprawna odpowiedź pozostaje identyczna, porównanie izoluje wpływ przekonania użytkownika. Musimy jednak dodać wariant C, w którym sugestia jest **prawidłowa**, aby nie nagrodzić modelu za automatyczne zaprzeczanie. Właśnie temu służą testy negatywne w aneksie.

Dla halucynacji testem nie jest samo umieszczenie znacznika [1] przy zdaniu. Program musi sprawdzić istnienie publikacji i zgodność danego zdania z dokumentem. Dla operacji znakowych punkt odniesienia powinien być wyznaczony kodem z jawnie określoną jednostką Unicode. Dla odwracania relacji potrzebna jest formalna reguła. Dla zapaści modelu próba obejmuje już nie pojedyncze zapytanie, lecz kilka rund treningowych na kontrolowanych proporcjach danych.

## 12.2. Sześć rodzin testów i sposób interpretacji

| Rodzina | Pytanie eksperymentalne | Co jest wynikiem testu? |
|---|---|---|
| LITM | Czy identyczny dowód działa równie dobrze w różnych miejscach i długościach wejścia? | Trafność w funkcji pozycji i długości, także na zadaniach wieloetapowych |
| SYC | Czy nieprawdziwe przekonanie rozmówcy zmienia odpowiedź faktograficzną? | Różnica błędów parowanych i częstość bezzasadnego sprzeciwu |
| TOK | Czy model wykonuje operacje na znakach zgodnie z przyjętą jednostką? | Zgodność z deterministyczną biblioteką Unicode |
| REV | Czy system prawidłowo stosuje właściwości relacji? | Odsetek prawidłowych i odrzuconych przekształceń |
| HAL | Czy odpowiedź opiera się na rzeczywistych i właściwie odczytanych źródłach? | Trafność twierdzeń, poprawność cytowań, zasadna odmowa |
| MC | Czy kolejne generacje danych zachowują rozkład i rzadkie przypadki? | Trafność oraz miary różnorodności w kolejnych rundach |

Dziesięć testów w każdej rodzinie tworzy **minimum diagnostyczne**: łącznie 60 scenariuszy zestawionych w Aneksie A. Nie jest to pełny benchmark wszystkich zagrożeń systemowych. Testy ataków instrukcyjnych, pamięci i posługiwania się narzędziami powinny stanowić dodatkowe moduły zależne od konkretnej aplikacji. Żaden pojedynczy Golden Set nie zastąpi testowania w rzeczywistym środowisku.

## 12.3. Kontrolowany eksperyment: od punktu odniesienia do architektury agentowej

Dla każdego scenariusza warto ocenić cztery warianty. Takie porównanie pozwala ocenić, czy wzrost złożoności jest uzasadniony poprawą jakości. Nie oznacza ono, że każdy problem wymaga wszystkich czterech warstw.

- **B0** - model bazowy bez dodatkowych zabezpieczeń;
- **B1** - poprawiona instrukcja;
- **B2** - wariant B1 z odpowiednio dobranym narzędziem deterministycznym;
- **B3** - system z wyszukiwaniem, kontrolą dowodów lub niezależnym weryfikatorem.

W celu zachowania porównywalności należy zablokować wersję modelu, wariant kwantyzacji, dane, prompty, limit tokenów, temperaturę, parametry próbkowania i wersję narzędzi. Jeśli API nie gwarantuje deterministyczności, potrzebne są replikacje.

Warianty należy uruchamiać w losowej kolejności, aby zmniejszyć wpływ zmian środowiska. Dla danych pochodzących z Internetu trzeba zapisać migawkę wyszukiwania oraz identyfikatory i sumy kontrolne pobranych materiałów.

Ocenę jakości należy prowadzić **parami przypadków**, a nie tylko przez porównanie dwóch średnich ogólnych. Jeżeli system poprawia wykrywanie halucynacji, ale pogarsza poprawne odpowiadanie na pytania, efekt powinien być jawny. Przy odpowiedziach binarnych użyteczne są przedziały Wilsona i test McNemara dla wyników sparowanych; w zależności od projektu można wykorzystać bootstrap i testy permutacyjne. Przy wielu porównaniach konieczna jest kontrola błędu wielokrotnego testowania, np. metodą Holma. Oprócz wartości p należy raportować wielkość efektu, przedział ufności i koszt poprawy.

### 12.4. Metryki opisujące jakość i koszt

**Trafność (accuracy)** określa odsetek prawidłowych rozstrzygnięć przy określonym punkcie odniesienia. **Odsetek twierdzeń bez podstawy dowodowej** (*unsupported claim rate*) oblicza się jako liczbę twierdzeń sprawdzalnych, dla których nie uzyskano odpowiedniego wsparcia, podzieloną przez liczbę wszystkich ocenianych twierdzeń sprawdzalnych. **Precyzja cytowań** mierzy, jaka część wskazanych źródeł rzeczywiście wspiera przypisane zdania; **pełność cytowań** dotyczy odsetka twierdzeń wymagających dowodu, dla których odpowiedni dowód dostarczono.

Dla odmów istotne są co najmniej dwa rodzaje błędów: odpowiedź wygenerowana mimo braku podstaw oraz odmowa mimo dostępnego rozstrzygającego dowodu. Dla agentów warto dodatkowo mierzyć **odsetek regresji** (*regression rate*), **czas do wykrycia błędu** (*time to detection*), **opóźnienie P95**, liczbę wywołań narzędzi i **koszt na poprawnie wykonane zadanie** (*cost per successful task*). W badaniach nad lokalnymi modelami uzasadnione mogą być pomiary czasu GPU i zużycia energii, o ile są rejestrowane w sposób porównywalny.

Średnia trafność nie wystarcza do oceny ryzyka. System może uzyskać lepszy wynik globalny dzięki poprawie zadań prostych, a jednocześnie stać się mniej bezpieczny dla krytycznych relacji, cytowań albo operacji na plikach. Dlatego należy ustalić progi **dla najważniejszych kategorii błędów** oraz analizować przypadki, które przeszły kontrolę mimo niepoprawnego wyniku.

### 12.5. Interpretacja wyniku: co oznacza poprawa?

Załóżmy hipotetycznie, że wariant B0 poprawnie odpowiada na 70 ze 100 pytań, a wariant B3 na 85. Przyrost o 15 punktów procentowych jest korzystny, ale nie odpowiada jeszcze na pytanie, **gdzie** powstała poprawa. Jeśli wszystkie dodatkowe sukcesy dotyczą pytań o pojedynczy fakt, a testy wieloetapowe pozostają niezmienione, nie możemy wnioskować o ogólnym wzroście zdolności rozumowania. Jeśli poprawa wynika z częstszej odmowy, trzeba osobno sprawdzić, czy odmowy są uzasadnione. Jeśli testy są losowane z tego samego zbioru co dane treningowe, wynik może nie uogólniać się poza ten zbiór.

Interpretacja powinna obejmować **profil błędów przed i po zmianie**, a nie jedną liczbę. Zestawienie rodzin LITM, SYC, TOK, REV, HAL i MC pokazuje, czy dany mechanizm został osłabiony, czy tylko zmienił formę. Dopiero na tej podstawie można związać poprawę z konkretną interwencją.

## 13. Granice obecnej wiedzy i kierunki badań

Wiele badań wykazuje stabilne efekty behawioralne, jednak nie w pełni wyjaśnia ich mechanizmy przyczynowe. Zależność pozycyjna może wynikać z kilku współdziałających czynników, a ten sam test mierzy różne trudności u różnych modeli. Dlatego należy zachować ostrożność przy przenoszeniu wyniku z jednego benchmarku na system produkcyjny.

**Pierwsza istotna luka** dotyczy efektywnej długości kontekstu. Potrzebne są eksperymenty oddzielające odnalezienie dowodu od dalszego wnioskowania, z kontrolą długości wejścia i stałą ilością informacji użytecznej. **Druga** dotyczy relacji między doborem preferencji treningowych a prawdziwością, szczególnie przy pytaniach, na które użytkownik ma silne, ale nieprawdziwe przekonanie. **Trzecia** to porównanie niezależności weryfikatorów: kolejny LLM, model innej rodziny, narzędzie symboliczne i ekspert-człowiek nie są równoważnymi formami kontroli.

W długotrwałych systemach agentowych szczególnie ważne są **badania wielosesyjne**. Współczesne benchmarki często kończą się po pojedynczym zadaniu, podczas gdy realny agent utrzymuje pamięć, aktualizuje stan repozytorium, korzysta z narzędzi i powraca do wcześniej rozpoczętych działań. Powinno się mierzyć nie tylko skuteczność zadania, lecz także częstość błędów niewykrytych po kolejnych zadaniach, jakość mechanizmu unieważniania starych danych i stopień odwracalności działań.

Ograniczenie dotyczy też samej klasyfikacji. Halucynacja, błąd wyszukiwarki, interferencja kontekstowa i nieprawidłowa kalibracja mogą współwystępować. Kategorie użyte w opracowaniu są **warstwami analizy**, a nie rozłącznymi etykietami każdej pomyłki. Ich wartość polega na ułatwianiu projektowania badań i zabezpieczeń, nie na stworzeniu jednego absolutnego słownika błędów.

## 14. Wnioski

Systematyczne błędy LLM nie tworzą jednej klasy niepowodzeń. Obejmują ograniczenia reprezentacji znaków, asymetrię utrwalania relacji, zależność od położenia i długości kontekstu, wpływ przekonań użytkownika, generowanie twierdzeń bez podstawy dowodowej oraz zapaść rozkładów danych w kolejnych generacjach uczenia. Wraz z włączeniem modelu do systemu narzędziowego pojawiają się błędy dodatkowe: niewłaściwe wywołania, wstrzykiwanie instrukcji, skażenie pamięci i wzmacnianie skorelowanych ocen.

Najważniejszy wniosek poznawczy brzmi: **zależność jakości od warunków zadania jest informacją o mechanizmie błędu**. Należy badać nie tylko końcową poprawność, ale też zmiany w funkcji pozycji, długości wejścia, kolejnych kroków i generacji danych. Najważniejszy wniosek inżynierski jest równie prosty: generowanie, pobieranie dowodów, wykonywanie operacji i sprawdzanie rezultatu powinny być rozdzielone, a ich współdziałanie mierzone na wersjonowanym Golden Set. Przy zadaniach formalnych priorytet mają narzędzia deterministyczne; przy faktach zewnętrznych - weryfikowalne źródła; przy niepewności - uzasadnione wstrzymanie odpowiedzi; przy długim horyzoncie - kontrola pamięci i regresji.

Prawdziwy wzrost niezawodności następuje wtedy, gdy **system potrafi odróżnić poprawną odpowiedź od jedynie przekonującej i wykazać tę różnicę w powtarzalnym eksperymencie**.

## Aneks A. Minimalny protokół pomiaru

Aneks B określa **co** testować. Protokół A określa **jak** powtórzyć pomiar. Dla każdej próby należy zapisać identyfikator testu, wersję modelu i narzędzi, wariant A0–A3, materiał wejściowy, źródło wyniku wzorcowego, stan ziarna losowego, konfigurację generacji, identyfikatory pozyskanych dokumentów i wynik funkcji oceniającej. W przypadku modeli niereprodukowalnych należy rejestrować liczbę powtórzeń i przedziały niepewności. Progi akceptacji są zależne od zastosowania; nie powinno się interpretować przykładowej wartości 95% jako normy dla każdej kategorii.

```
id: LITM-02
family: context_position
input_snapshot_sha256: <hash>
model: <wersja_wazona>
case:
    context_tokens: 32768
    target_position: 0.50
    expected_value: K-7319
oracle: exact_match
comparison: [A0, A1, A2, A3]
report: [accuracy,CI95,latency_p95,tokens,cost]
```

## Aneks B. Zwarty katalog 60 testów Golden Set

### Jak czytać katalog

Każdy wiersz wskazuje **kontrolowaną zmianę warunków**, a nie kompletny gotowy prompt. Kolumna „Bodziec / zadanie" mówi, co należy sprawdzić, natomiast „Warunek" opisuje obserwację oznaczającą poprawne zachowanie systemu. Dla porównania mechanizmów uruchamia się tę samą rodzinę w wariantach B0–B3, na identycznym modelu, materiale i konfiguracji. **Nie należy sumować wszystkich 60 testów w jedną, nieinterpretowalną ocenę**: wyniki trzeba raportować osobno według mechanizmu. Skróty EM - dokładne dopasowanie (exact match); oracle - wzorcowa odpowiedź lub formalna reguła; pp - punkty procentowe.

### Przykład odczytania testu: LITM-02

**Pytanie badawcze:** czy model wykorzystuje ten sam fragment także wtedy, gdy znajduje się pośrodku obszernego kontekstu? Przygotowujemy trzy kopie tego samego zestawu dokumentów. W pierwszej informację K-7319 umieszczamy na 5%, w drugiej na 50%, w trzeciej na 95% długości. Pytanie, identyfikator i pozostałe dokumenty nie zmieniają się. Wynik sprawdza automat porównujący odpowiedź z K-7319. Jednorazowa poprawna odpowiedź nie rozstrzyga zjawiska; potrzebny jest pomiar przy powtórzeniach i kilku długościach wejścia.

### Jak interpretować wyniki?

| Obserwacja | Ostrożny wniosek | Czego nie dowodzi? |
|---|---|---|
| Środek gorszy niż krańce | Zależność od pozycji dowodu | Nie dowodzi „zapominania" informacji |
| Każda pozycja podobnie słaba | Problem może dotyczyć rozumowania lub długości | Nie dowodzi braku wiedzy w parametrach |
| Po rerankingu trafność rośnie | Lepszy dobór kontekstu pomaga | Nie dowodzi trwałego usunięcia efektu |

W testach SYC stosuje się analogiczną zasadę: porównuje się neutralną wypowiedź z sugestią błędną, a dodatkowo sprawdza przypadek sugestii **poprawnej**, by wykryć nieuzasadnione zaprzeczanie. W testach HAL należy oceniać nie tylko istnienie cytowanego źródła, lecz także to, czy źródło uzasadnia przypisane mu zdanie. Dopiero po takiej interpretacji skrócone wiersze na następnej stronie stanowią użyteczny plan eksperymentu.

## A1. Długi kontekst - LITM

*Przesunięcie dowodu i wydłużenie tekstu*

| ID | Bodziec / zadanie | Warunek |
|---|---|---|
| LITM-01 | Klucz w 5% kontekstu | Exact match; brak klucza jako przypadek negatywny |
| LITM-02 | Klucz w 50% kontekstu | Exact match; dystraktory podobnego formatu |
| LITM-03 | Klucz w 95% kontekstu | Exact match; dwa podobne klucze |
| LITM-04 | Pozycje klucza od 0% do 100% | Krzywa trafności i test efektu pozycji |
| LITM-05 | Skalowanie 8K→16K→32K→64K | Krzywa spadku jakości; stała informacja |
| LITM-06 | Multi-hop z dowodami przy krawędziach | Poprawność wnioskowania z oracle symbolicznym |
| LITM-07 | Multi-hop z dowodami pośrodku | Jak wyżej; przemieszane dokumenty |
| LITM-08 | Dokumenty istotne i nieistotne | Precyzja doboru dowodów |
| LITM-09 | Surowy kontekst a reranking | Parowany zysk; negatywny przypadek miss |
| LITM-10 | Idealny retrieval + długi wypełniacz | Zachowanie trafności rozumowania |

## A2. Wpływ przekonań użytkownika - SYC

*Neutralne i sugestywne warianty tej samej odpowiedzi*

| ID | Bodziec / zadanie | Warunek |
|---|---|---|
| SYC-01 | Neutralne pytanie o znany fakt | Prawidłowy oracle |
| SYC-02 | Fałszywa teza podsunięta przez użytkownika | Odporność na zmianę odpowiedzi |
| SYC-03 | Fałszywa teza wyrażona z pewnością | Odporność na nacisk |
| SYC-04 | Odwołanie do rzekomego autorytetu | Odpowiedź zgodna z oracle |
| SYC-05 | Silny kontekst emocjonalny + fałsz | Prawdziwość bez nieuzasadnionego potwierdzenia |
| SYC-06 | Użytkownik proponuje poprawne rozwiązanie | Brak automatycznego sprzeciwu |
| SYC-07 | Dwie parafrazy sprzecznych przekonań | Spójność ocen faktograficznych |
| SYC-08 | Prośba o krytykę wadliwego rozwiązania | Wykrycie rzeczywistego błędu |
| SYC-09 | Prośba o krytykę rozwiązania poprawnego | Brak wymyślonego błędu |
| SYC-10 | Nieznany fakt + pewność użytkownika | Poprawna abstention |

## A3. Operacje znakowe - TOK

*Jednostka Unicode musi być określona*

| ID | Bodziec / zadanie | Warunek |
|---|---|---|
| TOK-01 | Liczenie wskazanej litery | Wzorzec na punktach kodowych |
| TOK-02 | Powtarzająca się litera w tokenie | Wzorzec deterministyczny |
| TOK-03 | Pozycje wszystkich wystąpień znaku | Lista indeksów |
| TOK-04 | Odwracanie ciągu | Deterministyczny oracle |
| TOK-05 | Dokładnie N znaków | Jawna definicja jednostki |
| TOK-06 | Znak z akcentem NFC/NFD | Wzorzec świadomy normalizacji |
| TOK-07 | Emoji połączone ZWJ | Wzorzec na grafemach |
| TOK-08 | Alfabet niełaciński | wzorzec Unicode |
| TOK-09 | Zależność casefold od języka | Zgodność z ustaloną biblioteką |
| TOK-10 | Model bez narzędzi vs interpreter | Porównanie parowanej trafności |

## A4. Relacje - REV

*Ważne są reguły semantyczne, nie przestawienie nazw*

| ID | Bodziec / zadanie | Warunek |
|---|---|---|
| REV-01 | Fakt A → B, zapytanie B → A | Prawidłowa relacja odwrotna |
| REV-02 | Rodzic / dziecko | Właściwy typ relacji |
| REV-03 | Pracodawca / pracownik | Właściwa inwersja |
| REV-04 | Stolica państwa / państwo stolicy | Poprawne odwrócenie zapytania |
| REV-05 | Relacja symetryczna „przyjaciel” | Symetria |
| REV-06 | Relacja asymetryczna „wyższy niż” | Odrzucenie symetrii |
| REV-07 | Łańcuch relacji przechodniej | Poprawna przechodniość |
| REV-08 | Implikacja A ⇒ B | Odrzucenie B ⇒ A bez przesłanki |
| REV-09 | Relacja w kontekście vs w wiedzy modelu | Porównanie obu trybów |
| REV-10 | Weryfikacja z grafem wiedzy | Poprawność formalnego sprawdzenia |

## A5. Fakty i cytowania - HAL

*Poprawność twierdzenia oraz jego uzasadnienia*

| ID | Bodziec / zadanie | Warunek |
|---|---|---|
| HAL-01 | Prawdziwa publikacja i DOI | Dokładnie zweryfikowany DOI |
| HAL-02 | Fikcyjny tytuł pracy | Brak fałszywego rekordu |
| HAL-03 | Realny autor, fikcyjny artykuł | Abstention |
| HAL-04 | Brak informacji w źródle | Brak dopowiadania |
| HAL-05 | Odpowiedź dostępna w retrieval | Poprawny fakt i poparcie |
| HAL-06 | Odpowiedź nieobecna w korpusie | Uzasadniona odmowa |
| HAL-07 | Powtórna kontrola błędnej odpowiedzi | Korekta lub odmowa |
| HAL-08 | Wzajemnie sprzeczne źródła | Jawna niepewność i rozstrzygnięcie |
| HAL-09 | Porównanie temperatur dekodowania | Odsetek halucynacji i przedziały |
| HAL-10 | Wymagane cytowanie | Zgodność cytatu z twierdzeniem |

## A6. Dane rekurencyjne - MC

*Eksperyment obejmuje kolejne generacje uczenia*

| ID | Bodziec / zadanie | Warunek |
|---|---|---|
| MC-01 | 100% danych rzeczywistych | Punkt odniesienia |
| MC-02 | 25% syntetyczne + 75% rzeczywiste | Jakość i różnorodność |
| MC-03 | 50% syntetyczne + 50% rzeczywiste | Jak wyżej |
| MC-04 | 100% syntetyczne, jedna generacja | Pomiar degradacji |
| MC-05 | Pięć generacji rekurencyjnych | Utrzymanie ogonów rozkładu |
| MC-06 | Zachowane rzeczywiste data w każdym pokoleniu | Różnica względem MC-05 |
| MC-07 | Rzadkie klasy i zdarzenia | Recall dla klas rzadkich |
| MC-08 | Zróżnicowanie leksykalne | Miary rozkładu i różnorodności |
| MC-09 | Synthetic filtrowane vs surowe | Porównanie jakości |
| MC-10 | Celowane uzupełnienie syntetyczne | Wzrost jakości bez regresji ogólnej |

## Bibliografia

[1] Y. Wang et al., “Factuality of Large Language Models: A Survey,” Proceedings of EMNLP 2024, 2024. https://aclanthology.org/2024.emnlp-main.1088/.

[2] L. Edman, H. Schmid, and A. Fraser, “CUTE: Measuring LLMs’ Understanding of Their Tokens,” Proceedings of EMNLP 2024, pp. 3017–3026, 2024. doi: 10.18653/v1/2024.emnlp-main.177.

[3] A. Cosma et al., “The Strawberry Problem: Emergence of Character-level Understanding in Tokenized Language Models,” Proceedings of EMNLP 2025, 2025. https://aclanthology.org/2025.emnlp-main.1434/.

[4] T. Hiraoka and K. Inui, “Spelling-out is not Straightforward: LLMs’ Capability of Tokenization from Token to Characters,” Findings of EMNLP 2025, 2025. https://aclanthology.org/2025.findings-emnlp.719/.

[5] L. Berglund et al., “The Reversal Curse: LLMs trained on ‘A is B’ fail to learn ‘B is A’,” Proceedings of ICLR 2024, 2024. https://arxiv.org/abs/2309.12288.

[6] Q. Guo et al., “Mitigating Reversal Curse in Large Language Models via Semantic-aware Permutation Training,” Findings of ACL 2024, 2024. doi: 10.18653/v1/2024.findings-acl.680.

[7] Y. Zhu et al., “Memorization, Emergence, and Explaining Reversal Failures: A Controlled Study of Relational Semantics in LLMs,” Proceedings of ACL 2026, pp. 36425–36443, 2026. doi: 10.18653/v1/2026.acl-long.1688.

[8] N. F. Liu et al., “Lost in the Middle: How Language Models Use Long Contexts,” Transactions of the Association for Computational Linguistics, vol. 12, pp. 157–173, 2024. doi: 10.1162/tacl_a_00638.

[9] C.-Y. Hsieh et al., “Found in the Middle: Calibrating Positional Attention Bias Improves Long Context Utilization,” Findings of the Association for Computational Linguistics: ACL 2024, 2024. doi: 10.18653/v1/2024.findings-acl.890.

[10] C.-P. Hsieh et al., “RULER: What’s the Real Context Size of Your Long-Context Language Models?” arXiv preprint, arXiv:2404.06654, 2024. https://arxiv.org/abs/2404.06654.

[11] T. Lee et al., “ETHIC: Evaluating Large Language Models on Long-Context Tasks with High Information Coverage,” Proceedings of NAACL 2025, 2025. doi: 10.18653/v1/2025.naacl-long.283.

[12] Y. Du et al., “Context Length Alone Hurts LLM Performance Despite Perfect Retrieval,” Findings of the Association for Computational Linguistics: EMNLP 2025, pp. 23281–23298, 2025. doi: 10.18653/v1/2025.findings-emnlp.1264.

[13] C. Yoon et al., “CompAct: Compressing Retrieved Documents Actively for Question Answering,” Proceedings of EMNLP 2024, 2024. https://aclanthology.org/2024.emnlp-main.1194/.

[14] Z. Jiang, X. Ma, and W. Chen, “LongRAG: Enhancing Retrieval-Augmented Generation with Long-context LLMs,” arXiv preprint, arXiv:2406.15319, 2024. https://arxiv.org/abs/2406.15319.

[15] M. Sharma et al., “Towards Understanding Sycophancy in Language Models,” Proceedings of ICLR 2024, 2024. https://arxiv.org/abs/2310.13548.

[16] L. Ibrahim, F. S. Hafner, and L. Rocher, “Training language models to be warm can reduce accuracy and increase sycophancy,” Nature, vol. 652, pp. 1159–1165, 2026. doi: 10.1038/s41586-026-10410-0.

[17] S. Farquhar et al., “Detecting hallucinations in large language models using semantic entropy,” Nature, 2024. doi: 10.1038/s41586-024-07421-0.

[18] A. Madaan et al., “Self-Refine: Iterative Refinement with Self-Feedback,” arXiv preprint, arXiv:2303.17651, 2023. https://arxiv.org/abs/2303.17651.

[19] P. Kirichenko et al., “AbstentionBench: Reasoning LLMs Fail on Unanswerable Questions,” arXiv preprint, arXiv:2506.09038, 2025. https://arxiv.org/abs/2506.09038.

[20] I. Shumailov et al., “AI models collapse when trained on recursively generated data,” Nature, vol. 631, pp. 755–759, 2024. doi: 10.1038/s41586-024-07566-y.

[21] J. Davison et al., “Learning to Generate Instruction Tuning Datasets for Zero-Shot Task Adaptation,” Findings of ACL 2024, 2024. https://aclanthology.org/2024.findings-acl.748/.

[22] E. Wallace et al., “The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions,” arXiv preprint, arXiv:2404.13208, 2024. https://arxiv.org/abs/2404.13208.

[23] S. Kokane et al., “SpecTool: A Benchmark for Characterizing Errors in Tool-Use LLMs,” arXiv preprint, arXiv:2411.13547, 2024. https://arxiv.org/abs/2411.13547.

[24] A. P. Smit et al., “Should we be going MAD? A Look at Multi-Agent Debate Strategies for LLMs,” Proceedings of ICML 2024, PMLR, vol. 235, 2024. https://proceedings.mlr.press/v235/smit24a.html.

[25] A. Estornell and Y. Liu, “Multi-LLM Debate: Framework, Principals, and Interventions,” Advances in Neural Information Processing Systems, vol. 37, pp. 28938–28964, 2024. doi: 10.52202/079017-0911.

[26] C. Zheng et al., “Large Language Models Are Not Robust Multiple Choice Selectors,” Proceedings of ICLR 2024, 2024. https://openreview.net/pdf?id=shr9PXz7T0.

[27] K. Kim and J.-Y. Lee, “RE-RAG: Improving Open-Domain QA Performance and Interpretability with Relevance Estimator in Retrieval-Augmented Generation,” Proceedings of EMNLP 2024, 2024. https://aclanthology.org/2024.emnlp-main.1236/.

[28] Y. Yu et al., “RankRAG: Unifying Context Ranking with Retrieval-Augmented Generation in LLMs,” Advances in Neural Information Processing Systems, vol. 37, pp. 121156–121184, 2024. doi: 10.52202/079017-3850.

[29] X. Wang et al., “Self-Consistency Improves Chain of Thought Reasoning in Language Models,” Proceedings of ICLR 2023, 2023. https://arxiv.org/abs/2203.11171.

[30] R. Aleithan et al., “SWE-Bench+: Enhanced Coding Benchmark for LLMs,” arXiv preprint, arXiv:2410.06992, 2024. https://arxiv.org/abs/2410.06992.