Kontekst i pamięć: co model wie, a czego nie

> Trzy różne magazyny informacji noszą w rozmowach o AI tę samą potoczną nazwę „pamięć", a mylenie ich jest źródłem większości nieporozumień w stylu „przecież mu to wcześniej pokazałem". Ta strona rozdziela je i pokazuje, gdzie każdy z nich się psuje.

Skróty użyte na tej stronie (5)

Skrót	Co znaczy
AI	Artificial Intelligence: sztuczna inteligencja, czyli programy naśladujące ludzkie rozumowanie
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
IT	Information Technology: dział albo praca związana z komputerami i infrastrukturą firmy
RAG	Retrieval-Augmented Generation: najpierw wyszukujemy w naszych dokumentach fragmenty pasujące do pytania, potem dopisujemy je do pytania zadawanego modelowi
RAM	Random Access Memory: pamięć operacyjna

Z życia wzięte

Z ŻYCIA WZIĘTE

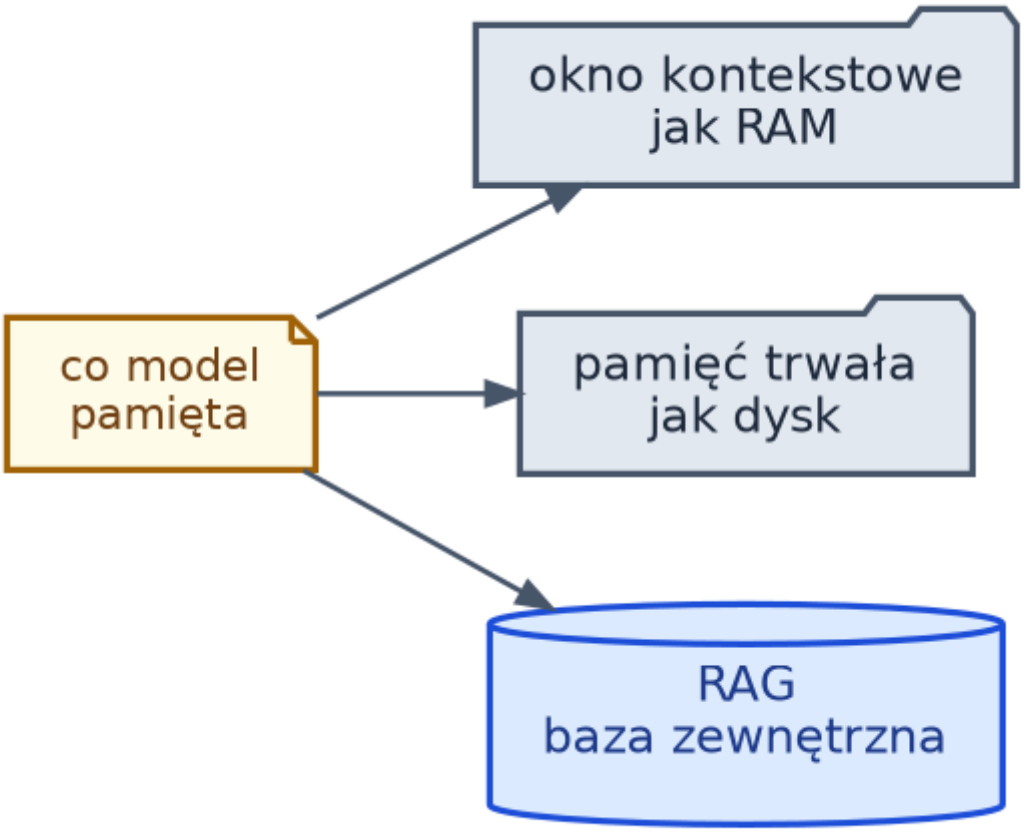
Pani Halina wkleja do czatu z Claude wyciąg z konta bankowego firmy, żeby dostać pomoc w rozpisaniu transakcji na kategorie przed zamknięciem miesiąca. Numery kont i salda trafiają w ten sposób do historii rozmowy, a stamtąd, jak każda wklejona treść, mogą powędrować do logów dostawcy usługi. Tłumaczymy jej, że lepiej wkleić sam wyciąg bez numerów kont albo poprosić kogoś z IT o przygotowanie zanonimizowanej wersji. Od tej pory pani Halina najpierw zamazuje numery w Wordzie, a dopiero potem wkleja resztę do czatu.

Trzy magazyny, nie jeden

Pojęcie: okno kontekstowe. Wszystko, co fizycznie trafia do pojedynczego wywołania modelu — instrukcja systemowa, historia rozmowy, wklejone pliki, wyniki narzędzi. Znika bezpowrotnie z końcem sesji albo w momencie, gdy przestaje się mieścić w limicie i coś musi zostać z niego usunięte. To odpowiednik pamięci RAM: szybki, dostępny od razu, ale nietrwały i ograniczony rozmiarem.

Pojęcie: pamięć trwała. Plik albo baza poza modelem, do której agent świadomie zapisuje fakty i którą odczytuje na starcie kolejnej sesji — konkretnym, dziś oficjalnym mechanizmem u Anthropic jest narzędzie pamięci (memory tool), które daje modelowi operacje na katalogu plików: podejrzenie zawartości, utworzenie, edycja fragmentu, wstawienie, usunięcie, zmiana nazwy. To odpowiednik dysku: wolniejszy w dostępie niż to, co już leży w oknie, ale przetrwa zamknięcie rozmowy i ma w zasadzie dowolny rozmiar, bo to my kontrolujemy, gdzie i jak długo jest przechowywany.

Pojęcie: baza wiedzy przeszukiwana na żądanie (RAG). Zewnętrzny magazyn treści — najczęściej baza wektorowa albo wyszukiwarka pełnotekstowa — z którego agent w trakcie zadania pobiera tylko fragmenty pasujące do bieżącego pytania, zamiast trzymać całą bazę w kontekście na stałe. RAG to skrót od Retrieval-Augmented Generation: generowanie odpowiedzi wspomagane wyszukiwaniem. W przeciwieństwie do pamięci trwałej treść w RAG zwykle nie jest pisana przez samego agenta, tylko przez osobny proces indeksujący dokumentację, wiki czy bazę zgłoszeń.



Trzy różne magazyny, które w rozmowie nazywa się jedną nazwą: pamięć (kliknięcie otwiera pełny rozmiar)

Rozróżnienie ma znaczenie praktyczne: pytanie „czy agent to pamięta" ma sens tylko wtedy, gdy wiemy, o który z trzech magazynów pytamy. Fakt ustalony dziesięć wiadomości temu w tej samej rozmowie żyje w oknie kontekstowym i zniknie z nowym wątkiem. Fakt zapisany świadomie do pliku pamięci przetrwa do następnej sesji. Fakt leżący w naszej dokumentacji firmowej będzie dostępny dopiero, gdy agent go wyszuka przez RAG albo gdy ktoś go wklei ręcznie.

Context engineering zamiast samego prompt engineering

> **Pojęcie: context engineering.** Dyscyplina świadomego doboru tego, co w danym momencie trafia do okna kontekstowego — nie tylko treści samej instrukcji, ale też tego, ile historii zostawić, które wyniki narzędzi zachować w całości, które streścić, a które odrzucić. O ile prompt engineering odpowiada na pytanie „jak sformułować polecenie", o tyle context

> engineering odpowiada na pytanie „co w ogóle powinno się znaleźć w polu widzenia modelu w tym konkretnym kroku" — i w agentach wieloetapowych to dziś większa część roboty niż samo układanie zdań polecenia, bo kontekst zmienia się przy każdym obrocie pętli, a nie raz na starcie rozmowy.

W praktyce oznacza to konkretne decyzje: czy wynik komendy `cat` całego pliku logów trafia do kontekstu w całości, czy najpierw przechodzi przez `grep`, żeby zostały tylko istotne linijki; czy pełna historia dwudziestu poprzednich kroków agenta jedzie dalej w każdym kolejnym wywołaniu, czy zostaje streszczona; które fakty warto od razu przenieść do pliku pamięci trwałej, żeby nie trzeba było ich trzymać w oknie przez resztę sesji.

Cztery sposoby, na jakie kontekst szkodzi sam sobie

> **Pojęcie: zatruwanie kontekstu (context poisoning).** Sytuacja, w której błędna informacja albo halucynacja modelu, raz zapisana w historii rozmowy, zostaje potem przez ten sam model potraktowana jako ustalony fakt i budowana na niej kolejna, jeszcze bardziej błędna odpowiedź. Agent, który zmyślił nazwę parametru w piątym kroku, w dziesiątym kroku odwołuje się do tej nazwy jak do czegoś sprawdzonego, bo z jego perspektywy „przecież to już ustaliliśmy" — leży to w kontekście, więc jest traktowane jak fakt.

> **Pojęcie: rozmycie uwagi w długim kontekście (context rot).** Mierzalny spadek jakości odpowiedzi modelu wraz ze wzrostem długości wejścia, widoczny znacznie wcześniej, niż okno kontekstowe fizycznie się zapełni. Model musi jednocześnie odnaleźć istotny fragment w zaszumionym tekście i rozumować na jego podstawie — im więcej nieistotnego materiału po drodze, tym gorzej idzie oba te zadania naraz. Do tego dochodzi efekt pozycji: informacja umieszczona na samym początku albo samym końcu długiego kontekstu bywa wykorzystywana lepiej niż ta sama informacja zakopana gdzieś pośrodku.

Dwa pozostałe tryby, o których warto wiedzieć, choć rzadziej nazywa się je osobno: rozproszenie uwagi przez nadmiar nieistotnych, ale tematycznie pasujących fragmentów (im więcej takich rozpraszaczy, tym gorszy wynik, nawet jeśli żaden z nich wprost nie wprowadza w błąd) oraz konflikt wewnątrz kontekstu, gdy dwa fragmenty historii mówią sobie wzajemnie zaprzeczające rzeczy i model musi wybrać, któremu zaufać.

Kompaktowanie rozmowy

>

> **Pojęcie: kompaktowanie (compaction).** Mechanizm, w którym długa historia rozmowy zostaje w pewnym momencie automatycznie streszczona i podmieniona na krótszą wersję, żeby zwolnić miejsce w oknie kontekstowym bez kończenia sesji. Zamiast dwudziestu tysięcy tokenów pełnej historii dwudziestu kroków agent dostaje kilka tysięcy tokenów streszczenia plus najświeższe kroki w pełnej treści. Cena tej operacji jest konkretna: streszczenie z definicji gubi szczegóły, a jeśli akurat ten pominięty szczegół był istotny dla dalszych kroków, kompaktowanie samo staje się źródłem błędu — swego rodzaju kontrolowana wersja zatrucia kontekstu, tyle że wprowadzona świadomie zamiast przez pomyłkę modelu.

Kiedy RAG ma sens, a kiedy wystarczy wkleić plik

Wklejenie pliku bezpośrednio do kontekstu wygrywa, gdy dokumentów jest niewiele, całość mieści się wygodnie w oknie razem z resztą zadania, a treść nie zmienia się w trakcie pracy — jednorazowa analiza jednego pliku konfiguracyjnego, jednego release notes, jednej procedury. Prostota tego podejścia jest realną zaletą: nie ma nic do zaindeksowania, nic do utrzymania, a model widzi całość naraz, bez ryzyka, że wyszukiwarka nie trafi w odpowiedni fragment.

RAG zaczyna się opłacać, gdy baza wiedzy jest zbyt duża, żeby zmieściła się w oknie razem z resztą zadania, gdy zmienia się częściej niż raz na sesję, albo gdy potrzebujemy wskazać źródło konkretnego fragmentu odpowiedzi, a nie tylko samą odpowiedź — cała nasza dokumentacja wewnętrzna, historia zgłoszeń helpdesku z ostatnich pięciu lat, baza wiedzy produktowej aktualizowana codziennie. To jest dziś obszar wyraźnie sporny: rosnące okna kontekstowe sprawiły, że część zadań, które kilka lat temu wymagały RAG, dziś rozwiązuje się prostym wklejeniem całego korpusu, a analizy z 2026 roku pokazują, że dla średniej wielkości baz sam długi kontekst bywa dokładniejszy niż RAG z niedoskonałym wyszukiwaniem, które wprowadza własny szum w postaci nietrafionych fragmentów. Z drugiej strony przy naprawdę dużych, zmiennych bazach RAG wciąż wygrywa kosztem — nie trzeba płacić za ponowne przetworzenie całej bazy w każdym zapytaniu — i kontrolą nad tym, co dokładnie trafiło do odpowiedzi. Nie ma tu jednej reguły ważnej na zawsze, tylko pytanie o rozmiar, zmienność i potrzebę cytowania źródła w naszym konkretnym przypadku.

Higiena: co wolno wkleić, a czego nie

Wolno i warto wklejać: fragmenty faktycznie potrzebne do zadania, dane już zanonimizowane albo niezawierające sekretów, wyniki komend przefiltrowane do istotnej treści.

Nie wolno wklejać: haseł, kluczy API, tokenów dostępowych i innych sekretów — nawet w prywatnej rozmowie, bo trafiają do historii, którą ktoś kiedyś może przejrzeć, wyeksportować albo która trafi do logów dostawcy usługi. Nie warto też wklejać całych, nieprzefiltrowanych logów czy zrzutów bazy zamiast ich istotnego wycinka — nie tylko z powodu kosztu w tokenach, ale właśnie z powodu rozmycia uwagi opisanego wyżej: więcej szumu w kontekście, gorsza odpowiedź, nie lepsza.

□□ Ćwiczenia

> Zadania sprawdzające.

1. Agent w dwudziestym kroku długiego zadania nagle zaczyna odwoływać się do nazwy zmiennej środowiskowej, której nikt wcześniej nie definiował ani nie widział w żadnym pliku. Które ze zjawisk opisanych na tej stronie najpewniej za tym stoi?

Odpowiedź

Zatruwanie kontekstu — model prawdopodobnie zmyślił tę nazwę w którymś z wcześniejszych kroków, a ponieważ ta zmyślona nazwa trafiła do historii rozmowy, kolejne kroki traktują ją jako ustalony fakt i budują na niej dalej.

2. Mamy do przeanalizowania jeden plik konfiguracyjny liczący dwieście linii. Czy warto budować dla niego indeks RAG?

Odpowiedź

Nie — dwieście linii mieści się bez trudu w oknie kontekstowym razem z resztą zadania, treść jest statyczna na czas analizy, a budowanie indeksu dla pojedynczego, małego pliku dokłada koszt utrzymania bez żadnej korzyści. Wklejenie pliku wprost jest tu prostsze i dokładniejsze.

3. Sesja trwa już półtorej godziny, historia rozmowy właśnie została automatycznie skompaktowana. Dlaczego to dobry moment, żeby sprawdzić, czy jakiś istotny szczegół ustalony wcześniej nie zniknął z odpowiedzi agenta?

Odpowiedź

Bo kompaktowanie z definicji jest streszczeniem, a streszczenie z definicji gubi szczegóły — jeżeli pominięty fragment był istotny dla dalszych kroków, agent będzie działał dalej tak, jakby ten szczegół nigdy nie istniał, bez żadnego ostrzeżenia, że coś zniknęło.

Źródła: Effective Context Engineering for AI Agents, Anthropic: https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents — dokumentacja narzędzia pamięci, Claude Platform: https://platform.claude.com/docs/en/agents-and-tools/tool-use/memory-tool — Context Rot: jak długość wejścia wpływa na jakość odpowiedzi, badanie Chroma (2025) i jego omówienia z 2026: https://www.tinyfish.ai/blog/context-rot — dyskusja RAG kontra długi kontekst, stan na 2026: https://usewire.io/blog/long-context-vs-rag-what-the-data-shows/

Uwaga: to jeden z najszybciej zmieniających się obszarów w całej książce — granica opłacalności RAG wobec długiego kontekstu przesuwa się z każdą kolejną generacją modeli, więc konkretne progi warto weryfikować na bieżąco, a nie traktować jako stałe.

Revision #35
Created 2026-09-25 10:33:16 UTC by Tomasz Konieczny
Updated 2026-10-26 14:10:11 UTC by Tomasz Konieczny