P A M I Ę Ć B T T F

Koniec ery agentów z amnezją

Bezstratna pamięć projektu w KreionAI

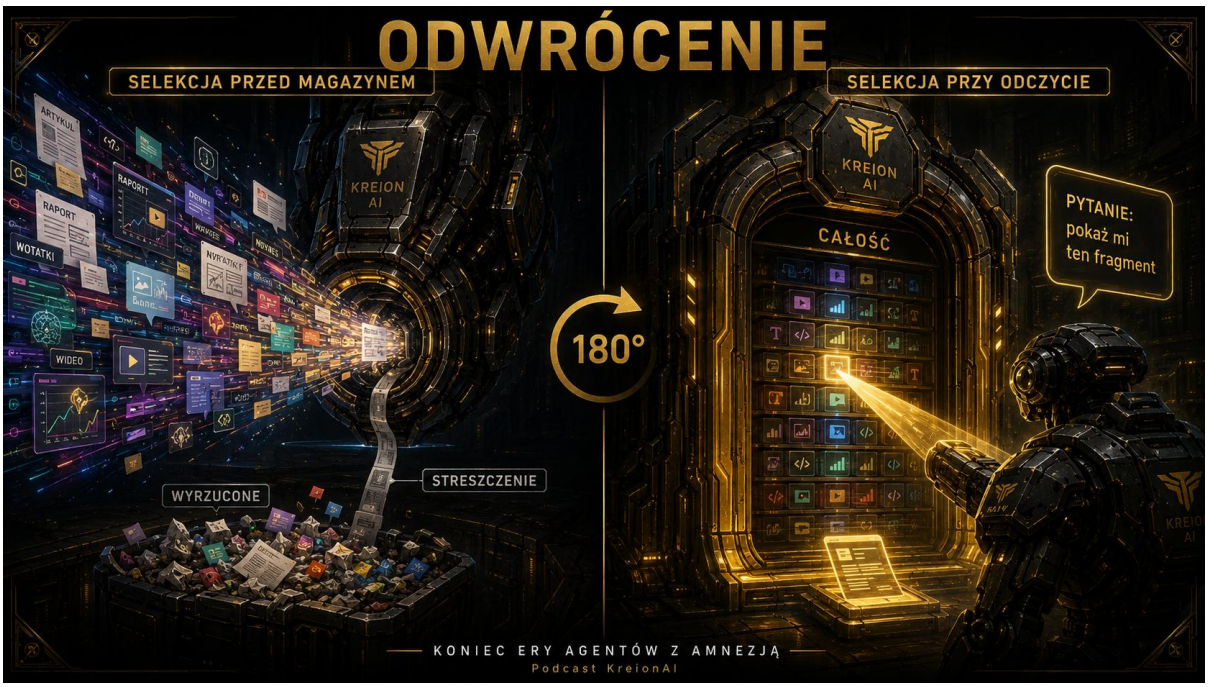
Mirosław Kardaś | Atnel

ZGODA, KTÓREJ NIKT NIE PODPISAŁ

Cała dzisiejsza praca z AI stoi na jednym cichym założeniu, które wszyscy przyjęli, choć nikt się na nie nie umawiał: agent zapomina. Kończysz sesję - wiedza umiera. Wracasz rano - briefujesz od zera. Zmieniasz narzędzie albo dostawcę modelu - zostawiasz całą wspólną historię za sobą, jakby jej nigdy nie było.

Branża nie traktuje tego jak wady. Traktuje to jak prawo natury. Dlatego buduje coraz lepsze protezy: kompaktowanie kontekstu, automatyczne streszczenia, notatki pamięci, indeksy wektorowe. Każda z tych protez robi w istocie to samo - w momencie zapisu DECYDUJE, co będzie ważne, a resztę wyrzuca. Streszczenie to decyzja. Embedding to decyzja. Kompaktacja to decyzja. A każda taka decyzja jest podejmowana w najgorszym możliwym momencie: zanim wiadomo, o co za trzy miesiące zapytasz.

Informacja wyrzucona przy zapisie nie wraca nigdy. Żadne późniejsze wyszukiwanie jej nie odzyska, bo jej fizycznie nie ma. Sufit wierności pamięci zostaje obniżony raz i na zawsze - i to jest fundament, na którym stoi dziś praktycznie każde narzędzie agentowe na rynku.



ODWRÓCENIE

Pamięć BTTF w KreionAI odwraca ten fundament o sto osiemdziesiąt stopni: **zapis jest bezstratny, a selekcja następuje dopiero przy odczycie** - w momencie, gdy pytanie jest już znane. W terminologii, której świat będzie musiał się nauczyć: ***lossless capture*** zamiast selekcji przy zapisie, ***retrieval-time selection*** zamiast zgadywania przyszłej ważności.

Kartridż BTTF przechowuje całą współpracę od pierwszej sesji: każdą rozmowę słowo w słowo, każdą decyzję, każde ustalenie - i każdy załącznik w wersji 1:1. Screenshoty, PDF-y, schematy, arkusze pomiarów, źródła - dokładnie te bajty, które agent wtedy analizował. Niczego się nie streszcza, niczego nie wyrzuca, niczego nie zamienia na przybliżenie. Po pół roku pytanie nie brzmi "czy agent pamięta, że analizowaliśmy schemat" - pytanie brzmi "czy agent ma przed sobą dokładnie ten schemat, w tej wersji" - i odpowiedź brzmi: tak. Niezależnie od tego, co przez pół roku stało się z folderami, archiwami i repozytoriami.

To zmienia charakter samej historii. Zapis bez źródeł mówi tylko: "agent twierdził wtedy X". Zapis ze źródłami mówi: "agent twierdził X - a oto dokładny materiał, który wtedy widział, więc jego wniosek można dziś ponownie zweryfikować". To już nie jest pamięć - to pełna proweniencja pracy (***work provenance***): każdy wniosek w historii ma obok siebie swój materiał dowodowy. I konsekwencja, którą mało kto zauważa, a która czyni kartridż inwestycją zyskującą na wartości z czasem: modele mądrzeją z każdym rokiem. Przyszły, znacznie lepszy model może wrócić do tego samego schematu sprzed roku i zobaczyć w nim to, czego dzisiejszy nie dostrzegł (***re-analysis under future intelligence***). Streszczenie albo embedding zamraża na zawsze ograniczenia modelu, który je wytworzył - źródło zachowane 1:1 pozwala każdej przyszłej inteligencji wrócić do dowodu pierwotnego.

I rzecz, która odróżnia BTTF od wszystkiego, co nazywa się dziś "pamięcią AI": kartridż nie należy do sesji, nie należy do narzędzia i nie należy do dostawcy modelu. **Kartridż należy do projektu** (***project-owned memory***) - a modele są wobec niego tylko wymienną mocą obliczeniową (***replaceable compute***). Tak jak kartridż z grą nie należał do konsoli, na której akurat grałeś.

NA KTÓRYM PIĘTRZE TO DZIAŁA

Żeby uciąć nieporozumienie, które wraca w co drugiej technicznej dyskusji: problem "pamięci AI" ma dwa piętra i warto je rozdzielić raz na zawsze.

Piętro pierwsze to sam model: architektura transformera, okno kontekstu, mechanizm uwagi, degradacja przy długich kontekstach. Tego piętra KreionAI nie dotyka w ogóle i nigdzie nie twierdzi inaczej. Pracują tu pełnoprawni, oficjalni agenci - bez jednej zmiany w ich wnętrzu, na zwykłych, dzisiejszych modelach. Na tym piętrze niech się ścigają dostawcy.

Piętro drugie to system pracy: GDZIE żyje trwała wiedza współpracy człowieka z agentami, gdy sesja się kończy. I to jest piętro BTTF - w całości. Cała opisana tu bezstratność, ciągłość i przenośność dzieje się w warstwie architektury pracy i narzędzi, nie w warstwie AI.

I teraz najciekawsze: techniczni odbiorcy, słysząc o efektach BTTF, odruchowo pytają, czy to jakaś nowa architektura modeli - bo efekty brzmią im jak z następnej epoki AI. Nie, i właśnie to jest cała teza tego dokumentu: branża od lat szuka rozwiązania problemu pamięci na piętrze modelu - w większych oknach, lepszym mechanizmie uwagi, przyszłych architekturach - podczas gdy problem od początku był rozwiązywalny piętro wyżej. Nie trzeba było zmieniać AI. Trzeba było zmienić miejsce, w którym mieszka wiedza.

CO Z TEGO WYNIKA W PRAKTYCE

Powrót do pracy przestaje istnieć jako czynność. Otwierasz projekt po tygodniu, po miesiącu, po kwartale - i pytasz "na czym skończyliśmy". W kilka sekund dostajesz odpowiedź: co zrobione, co czeka, jakie decyzje zapadły i dlaczego. Zero briefowania, zero wklejania kontekstów, zero płacenia - czasem i pieniędzmi - za przypominanie agentowi, kim jesteś. Koszt odtwarzania kontekstu (***context reconstruction cost***) nie został tu zoptymalizowany - on przestał istnieć.

Sesje stają się jednorazowe i wymienne. Skoro ciągłość trzyma kartridż, sesja przestaje być czymś, co trzeba pielęgnować, chronić i sztucznie przedłużać. Degradacja długiego kontekstu - zmora każdego narzędzia sesyjnego - przestaje być frontem, na którym trzeba walczyć.

Zmieniasz model - nie zmieniasz partnera. Na jednym kartridżu pracują pełnoprawni, oficjalni agenci różnych dostawców, a każda tura w zapisie jest podpisana modelem, który ją wykonał. Pracę wykonaną przez agenta jednej firmy możesz oddać do audytu agentowi drugiej (***cross-model peer review***) - zbudowanemu inaczej, z innymi nawykami i innymi ślepymi plamkami - a on widzi nie streszczenie i nie sam wynik, tylko pełny materiał: co dokładnie zleciłeś, co zostało zrobione, na jakiej podstawie i przez kogo. Systematyczne błędy jednego modelu wyłapuje inaczej zbudowany drugi, zamiast powielać się w nieskończoność. Tego układu nie ma żaden inny produkt na rynku.

Praca staje się przekazywalna jak plik. Kartridż można komuś dać. Druga osoba otwiera go u siebie, jej agent wchodzi w projekt z pełnym zrozumieniem - łącznie z materiałem źródłowym i tokiem rozumowania - dokłada swoją pracę i oddaje. Spróbuj przekazać komuś "stan pracy" złożony z historii czatów, notatek, repozytorium, folderów z grafikami i archiwów. To jest wykonalne - i to jest projekt sam w sobie. Tu jest jednym plikiem: przenośną jednostką pracy (***portable unit of work***).

Projekt zna własną ekonomię. Kartridż jest też rejestrem wykonania (***execution ledger***): każda tura niesie swój rzeczywisty koszt i czas rozumowania modelu. Projekt ma pełną, audytowalną historię finansową - wiadomo, ile kosztowała każda faza pracy, który model ile myślał nad którym problemem i gdzie koszty rosły. Historia czatu w innych narzędziach odpowiada najwyżej na pytanie "co powiedzieliśmy". Kartridż odpowiada też na pytania "ile to kosztowało" i "kto to wykonał".

ARCHEOLOGIA DECYZJI - NAJDROŻSZE PYTANIE W INŻYNIERII

Jest jedno pytanie, które kosztuje zespoły inżynierskie więcej niż jakiekolwiek inne, a które prawie nigdy nie pada w testach narzędzi AI: **"dlaczego wtedy podjęliśmy taką decyzję?"**

Historia rozwoju każdego poważnego projektu bywa kręta. Koncepcje ścierają się ze sobą, decyzje zapadają i bywają odwracane, a po miesiącach potrafi się okazać, że porzucona na początku droga była jednak lepsza - i trzeba do niej wrócić. Tylko na jakiej podstawie? Repozytorium kodu tu nie pomoże: commit zapisuje, CO zmieniono, gałąź zapisuje, ŻE czegoś próbowano - ale całe DLACZEGO, czyli argumenty, obawy, odrzucone alternatywy i powody rozstrzygnięć, nie mieszka w żadnym pliku. Ono padło w rozmowach. A rozmowy w każdym narzędziu na rynku umierają razem z sesją.

Nie trzeba wierzyć nam na słowo, że to jest problem pierwszej wielkości - branża opisuje go dziś sama, otwarcie i coraz głośniej. Deweloper prowadzący duży projekt z agentami pisze wprost, że najtrudniejsza nie jest utrata konwencji kodowania, tylko utrata wiedzy o tym, dlaczego projekt nie powinien ponownie skręcić w kierunek, który już kiedyś odrzucono: takie decyzje są nośne jak filary domu, a gdy znika powód ich istnienia, filar zaczyna wyglądać na opcjonalny. Ten sam autor stawia diagnozę, którą warto zapamiętać: znaczna część frustracji pracy z AI to w rzeczywistości awaria pamięci projektu przebrana za problem modelu. Inni dokumentują granicę plików kontekstowych typu CLAUDE.md: taki plik opisze architekturę, ale nie uchwyci wiedzy w rodzaju "wczoraj wybraliśmy podejście X, bo Y sprawiało problemy" - a to jest właśnie wiedza instytucjonalna, która znika z końcem sesji.

Jak branża próbuje ten problem rozwiązać? Wyłącznie ręczną dyscypliną. Wraca moda na Architecture Decision Records - formalne notatki o decyzjach, prowadzone przez ludzi - przy czym nawet ich zwolennicy przyznają otwarcie, że to najlepsza praktyka, której prawie nikt konsekwentnie nie utrzymuje, bo wymaga przerwania pracy i pisania dokumentacji "na zapas". Powstają prace naukowe o przerabianiu opisów commitów na protokół wiedzy dla agentów - czyli protezy budowane z repozytorium, bo zapisu rozmów nie ma - i te same prace przyznają, że formalne rekordy łapią co najwyżej decyzje architektoniczne, podczas gdy ginie masa decyzji implementacyjnych, których jest wielokrotnie więcej. Wspólny mianownik wszystkich tych podejść:

człowiek albo agent musi W MOMENCIE ZAPISU rozpoznać, że coś będzie ważne, i zdążyć to zanotować.

I tu jeden z autorów tej dyskusji, opisując własne zmagania z systemem notatek, sformułował zdanie, które rozstrzyga cały spór - nie znając rozwiązania, które właśnie czytasz: **"ważność jest retrospektywna"**. Notatka, która wczoraj wyglądała na jednorazowy szum, trzy sesje później okazuje się decyzją, która wyjaśnia wszystko. To zdanie jest wyrokiem na całą architekturę selekcji przy zapisie: skoro nie da się z góry wiedzieć, co okaże się ważne, to każdy system, który przy zapisie wybiera i wyrzuca, jest błędem konstrukcyjnym - niezależnie od tego, jak sprytnie wybiera. Istnieje dokładnie jedna architektura zgodna z faktem retrospektywnej ważności: **zachować wszystko, a wybierać dopiero wtedy, gdy pytanie jest znane.** Branża, nie znając BTTF, wyprowadziła publicznie jego uzasadnienie.

A teraz różnica w praktyce. W KreionAI takie pytanie zadaje się jednym zdaniem, naturalnym językiem, bez żadnych identyfikatorów i bez wskazywania, gdzie szukać: "powiedz mi, na czym polegała tamta duża zmiana, dlaczego ją zrobiliśmy i co było w niej najważniejsze". Odpowiedź przychodzi w minutę i zawiera to, czego nie ma w żadnym pliku na dysku: pierwotną przyczynę z datami, historię awarii, które wymusiły zmianę, przebieg dyskusji - łącznie ze sporem koncepcyjnym, w którym propozycja agenta przegrała z propozycją użytkownika, co agent sam odnotował i uczciwie przywołał - oraz zasadę, którą całość zamknięto, ze wskazaniem źródła w historii współpracy. Odrzucone koncepcje, argumenty, które przegrały, powody rozstrzygnięć: pełna stratygrafia decyzji, dostępna od ręki, bez jednej notatki prowadzonej przez człowieka. W języku, którym branża opisuje swoje marzenia, to jest ***decision provenance*** - pełne pochodzenie każdej decyzji wraz z odrzuconymi alternatywami i danymi, na których rozstrzygano - oraz pamięć instytucjonalna (***institutional memory***) powstająca w trakcie normalnej pracy, a nie jako osobny proces dokumentacyjny, którego i tak nikt nie utrzymuje.

To nie jest funkcja pokazowa. To jest odpowiedź na pytanie, na którym od dekad wykłada się każdy zespół inżynierski świata - i które dotąd wymagało albo pamięci ludzi, którzy odeszli, albo dokumentacji, której nikt nie napisał. Czytelniku, test jest prosty: zadaj swojemu narzędziu pytanie "dlaczego my to w ogóle zrobiliśmy?" o decyzję sprzed dwóch miesięcy - i porównaj.

DLACZEGO NIKT TEGO PO PROSTU NIE SKOPIUJE

Bo to nie jest funkcja - to jest fundament. Cursor, Antigravity, opencode, Hermes i cała reszta stoją na architekturze, w której pamięć projektu jest streszczana przy zapisie albo przywiązana do sesji i instalacji narzędzia - nigdy nie jest własnością projektu z pełnym materiałem źródłowym. Bezstratna pamięć jako własność projektu wymagałaby od nich nie dopisania modułu, tylko przebudowy fundamentów, na których stoi wszystko powyżej. Wielki gracz z milionami użytkowników nie zawraca z takiej drogi. Pojedyncze elementy będą naśladowane - i dobrze, bo to potwierdzi kierunek. Całości nie da się dokleić.

SYMBIOZA, NIE ZAMIENNIK

Jedno wyjaśnienie dla czytelników, którzy w tym miejscu szukają już tylko odpowiedzi na pytanie "czyli co - koniec RAG-ów i wektorów?". Nie. BTTF niczemu nie wypowiada wojny - i można to powiedzieć znacznie mocniej, bo tu leży ostateczna odpowiedź na wszystkie spory tego rodzaju: **BTTF bez tamtego świata w ogóle by nie istniał.** Żywi się pracą agentów, stoi na dzisiejszych modelach i korzysta z dorobku mechanizmów wyszukiwania i selekcji, które świat AI wypracowywał latami - a nadbudowuje nad nimi piętro, którego dotąd brakowało. Zależność jest przy tym jednokierunkowa: dotychczasowe rozwiązania - RAG, embeddingi, indeksy wektorowe - mogą istnieć bez BTTF i istniały na długo przed nim. BTTF bez nich nie istnieje. Trudno oskarżać o wypowiedzenie wojny coś, co otwarcie deklaruje, że bez drugiej strony nie ma racji bytu. Słowo, które opisuje ten układ najtrafniej, brzmi: **symbioza**. Spór "BTTF czy RAG" jest więc źle postawiony - to tak, jakby pytać "fundament czy winda". Budynek może mieć jedno i drugie; bez fundamentu winda nie ma tylko na czym stanąć.

A jeśli potrzebujesz obrazu, który zostanie w głowie na dłużej niż każda specyfikacja - oto on. BTTF jest jak huba na drzewie. Drzewa - sesje agentów - rosną, pracują i padają; taka jest ich rola i nie ma czego żałować. Huba zostaje. Karmi się każdym drzewem, które przeżyła, i z każdym kolejnym rośnie w siłę. I zwróć uwagę na ścisłość tego obrazu: huba nie istnieje bez drzew, a las radził sobie bez huby od zawsze - a jednak to ona jest tym, co pamięta las. A gdy przychodzi pora, robi to, do czego hubę wykorzystywano od tysięcy lat: staje się hubką pod krzesiwo. Od niej, jedną iskrą, zapala się każda nowa, pusta sesja - i w minutę płonie pełnym ogniem wiedzy całego projektu. Wartość nigdy nie mieszkała w drzewie. Wartość mieszka w tym, co zostaje, gdy drzewa już nie ma.

NAJPIERW DZIWNE, POTEM OCZYWISTE

Uczciwie: do BTTF trzeba się przyzwyczaić. Pierwsze dni są nieintuicyjne, bo lata pracy z narzędziami sesyjnymi wyrobiły odruchy: człowiek automatycznie briefuje agenta, przypomina mu ustalenia, wkleja konteksty "na wszelki wypadek", streszcza własny projekt po raz dwudziesty. Aż w którymś momencie orientuje się, że wykonuje pracę, która nie jest już potrzebna. Że można po prostu zapytać "dlaczego w marcu odrzuciliśmy tamto rozwiązanie" - i dostać odpowiedź.

A potem następuje drugi moment, ważniejszy: próba powrotu do jakiegokolwiek narzędzia sesyjnego. I nagle tamten świat wygląda tak, jak naprawdę wyglądał zawsze - jak praca ze zdolnym współpracownikiem, który każdego ranka ma całkowitą amnezję i któremu od nowa trzeba tłumaczyć, kim jesteś, nad czym pracujecie i co ustaliliście wczoraj.

Tak działała każda zmiana tej rangi. Autozapis był "zbędny", dopóki nie stał się oczywisty. Cofanie zmian było "luksusem", dopóki nie stało się powietrzem. Pamięć, która nie umiera z sesją, przechodzi teraz dokładnie tę samą drogę - od "po co to komu" do "jak można było pracować inaczej".

Różnica między narzędziami AI przestaje przebiegać tam, gdzie wszyscy ją mierzą - między modelami, oknami kontekstu i listami funkcji. Zaczyna przebiegać między dwiema formami współpracy: wykonawcą na godziny, który jutro cię nie pozna, a partnerem, który pamięta wszystko od pierwszego dnia.

BTTF to jest ta druga forma. I z niej się już nie wraca.