Pracownicy naukowi i zaprzyjaźnione osoby zewnętrzne

MODEL JĘZYKOWY · INŻYNIERIA · SOLVERY

Stanisław

Polski model inżynierski w praktyce

Przewodnik po budowie, pracy i rozwoju modelu, który liczy solverami, a nie zgaduje.



OD ZERA · U SIEBIE · Z SOLVERAMI

Wydział Wiertnictwa, Nafty i Gazu AGH
Niekomercyjny projekt badawczy
Wydanie pierwsze, robocze · Kraków, listopad 2026

WYDZIAŁ WIERTNICTWA, NAFTY I GAZU AGH

Stanisław

Polski model inżynierski w praktyce

Od zera, w zamkniętej infrastrukturze, z solverami.
Przewodnik po budowie, pracy i rozwoju

Pracownicy naukowi i zaprzyjaźnione osoby zewnętrzne

Wydanie pierwsze, robocze · Kraków, listopad 2026

Stanisław. Polski model inżynierski w praktyce

Od zera, w zamkniętej infrastrukturze, z solverami. Przewodnik po budowie, pracy i rozwoju

Pracownicy naukowi i zaprzyjaźnione osoby zewnętrzne. Wydanie pierwsze, robocze · Kraków, listopad 2026.

Docelowo Stanisław ma być wszechstronnym modelem inżynierskim. Zaczynamy od wiedzy, na której znamy się najlepiej: od technologii pozyskiwania węglowodorów.

Stanisław jest niekomercyjnym projektem badawczym. Model nosi imię prof. Stanisława Stryczka (1946-2024), wieloletniego dziekana Wydziału. Liczby w tej książce pochodzą z pomiarów zapisanych w repozytorium projektu, stan na 12 listopada 2026 r. Tam, gdzie wyniku jeszcze nie ma, piszemy o tym wprost.

Książka powstała we współpracy z asystentami AI, którzy pisali i sprawdzali tekst pod kierunkiem autorów. Każde twierdzenie o projekcie ma wskazane źródło w dokumentacji projektu.

Okładka: rysunek liniowy wykonany generatorem obrazów na podstawie szkicu portretu prof. Stanisława Stryczka, który powstał z fotografii z serwisu Historia AGH.

Strona projektu, dane na żywo z treningu i pełna instrukcja odtworzenia: stanislaw.info

Spis treści

	Wstęp	5
1	Dlaczego własny model inżynierski	7
2	Od zera: co to znaczy i ile kosztuje	11
3	Zamknięta infrastruktura	18
4	Solvery jako organy modelu	22
5	Jak pracować ze Stanisławem	29
6	Wiedza z dokumentów	34
7	Skąd wiemy, że działa	38
8	Odtworzenie krok po kroku w skrócie	43
9	Plan: 3B, 7B i dalej	47
10	Własny model w twojej organizacji	52
	Słowniczek	55
	Patron: Profesor Stanisław Stryczek	58

Wstęp

Dla kogo

Piszemy dla inżynierów i kierowników z firm technicznych, dla pracowników naukowych i dla studentów ostatnich lat. Specjalność nie ma znaczenia: mechanik, budowlaniec, energetyk i inżynier naftowy mają ten sam kłopot. Zakładamy, że czytelnik zna inżynierię, a uczenia maszynowego nie zna. Ma za to dokumenty, których nie wolno wynieść poza firmę albo uczelnię: raporty z badań, projekty, normy i podręczniki kupione na licencji. Chce wiedzieć, czy własny model językowy jest dla niego i ile pracy kosztuje.

Problem, który znamy wszyscy

Model językowy pisze płynnie, ale liczy z prawdopodobieństwa, a nie z rachunku. Potrafi podać wynik, który wygląda wiarygodnie i jest błędny. Potrafi też z równą pewnością opisać rzecz, której nie ma w żadnym dokumencie. W pracy inżyniera oba błędy są gorsze od braku odpowiedzi. Asystent, któremu można zaufać, powinien więc nie liczyć sam, znać dokumenty i umieć powiedzieć „nie wiem".

Czym jest Stanisław

Stanisław to polski model językowy, który budujemy na Wydziale Wiertnictwa, Nafty i Gazu AGH. Nosi imię prof. Stanisława Stryczka, w latach 2002-2008 dziekana tego Wydziału. Docelowo ma być wszechstronnym modelem inżynierskim. Zaczynamy od wiedzy, na której znamy się najlepiej: od technologii pozyskiwania węglowodorów. Katalog obliczeń i solvery dotyczą na razie wierceń i tak je w książce nazywamy. Baza dokumentów obejmuje szerzej naftę, gaz i nauki o Ziemi.

Model powstaje od zera. Korpus złożyliśmy sami z publicznych zbiorów tekstów, artykułów z czasopism i książek branżowych, a powtórzenia między źródłami i teksty pokrywające się z testami usunęliśmy własnym kodem. Tokenizer, czyli słownik kawałków, na które model tnie tekst, wytrenowaliśmy na własnej próbce. Wagi, czyli liczby, w których model przechowuje to, czego się nauczył, zaczynały od wartości losowych. Nie dostrajamy więc cudzego modelu. Pretrening, strojenie, wagi i praca Stanisława odbywają się na serwerach, z których korzystamy bezpośrednio. Przy pisaniu kodu i przeglądzie danych pomagali nam asystenci AI działający jako usługi zewnętrzne; piszemy o tym w rozdziale 3.

Najważniejsza zasada projektu brzmi: liczby liczy solver. Solver to deterministyczny program obliczeniowy, który dla tych samych danych zawsze daje ten sam wynik. Gdy ktoś pyta o ciśnienie słupa płuczki wiertniczej o gęstości 1200 kg/m³ na głębokości 2000 m, model ma rozpoznać rodzaj obliczenia, przepisać z pytania obie liczby i wywołać solver. Wynik 23,54 MPa podaje program, a model tylko go opisuje. Płuczka to ciecz krążąca w otworze podczas wiercenia, ale mechanizm byłby ten sam dla naprężenia w belce czy spadku napięcia w kablu.

Wydanie robocze

To pierwsze, robocze wydanie. Liczby pochodzą z pomiarów zapisanych w repozytorium projektu i odpowiadają stanowi z 11-12 listopada 2026 r. Model 1B, o miliardzie wag, jest gotowy. Model 3B jest w trakcie pretreningu, czyli pierwszego i najdłuższego etapu uczenia na surowym tekście, więc części wyników jeszcze nie ma i tak to zaznaczamy. Piszemy też o tym, co nie działa. Nasz model 1B w polskim teście rozumienia czytanego tekstu Belebele, w formacie wyboru odpowiedzi A-D, wypada na poziomie zgadywania. Bez takich zdań trudno byłoby ufać pozostałym liczbom.

Stanisław jest niekomercyjnym projektem badawczym Wydziału. Pełna instrukcja odtworzenia modelu, etap po etapie, z komendami i znanymi pułapkami, jest na stronie stanislaw.info.

ROZDZIAŁ 1

Dlaczego własny model inżynierski

Problem wspólny dla całej inżynierii

Model językowy uczy się przewidywać kolejny fragment tekstu na podstawie tego, co widział w treningu. Liczbę wyniku odtwarza z prawdopodobieństwa, nie z rachunku. Fakt, którego nie zna, potrafi uzupełnić czymś, co tylko brzmi prawdopodobnie. Dla autora listów to drobna wada, dla inżyniera dyskwalifikująca. Projektant konstrukcji, energetyk i technolog potrzebują od asystenta tego samego: żeby nie liczył z głowy, żeby mówił na podstawie dokumentów i żeby przyznał, kiedy odpowiedzi nie zna.

Drugi kłopot jest organizacyjny. Dokumenty, na których pracuje inżynier, rzadko są publiczne. To raporty z badań, projekty, normy i podręczniki kupione na licencji. Model, który ma z nich korzystać, musi działać tam, gdzie te dokumenty legalnie leżą.

Model inżynierski, który zaczyna od węglowodorów

Docelowo Stanisław ma być wszechstronnym modelem inżynierskim. Zaczynamy od technologii pozyskiwania węglowodorów, bo na tej wiedzy znamy się najlepiej. Wydział ma podręczniki, zadania z kolokwiów i naukowców, którzy sprawdzą odpowiedź modelu. Katalog obliczeń i solvery dotyczą dziś wierceń, a baza dokumentów obejmuje szerzej naftę, gaz i nauki o Ziemi. Przykłady w książce pochodzą więc z naszego podwórka, ale metoda od dziedziny nie zależy.

Dane chronione prawem

Najsilniej chronione teksty w naszym korpusie to książki branżowe: 210 książek w postaci PDF i 59 skanów odczytanych programem do rozpoznawania tekstu, razem 32 mln znaków po polsku. Są chronione prawem autorskim. Korzystamy z nich na podstawie art. 26^2 prawa autorskiego, który dopuszcza eksplorację tekstów i danych do badań naukowych. Samych tekstów nie publikujemy i nie przekazujemy nikomu jako danych do uczenia. Przy budowie katalogu wzorów fragmenty książek czytali jednak asystenci AI, z których korzystaliśmy przy pracy.

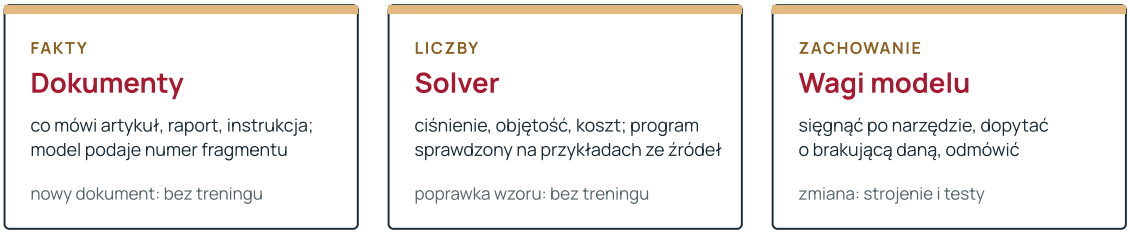
Z normami jest jeszcze ciaśniej. Licencje części wydawców norm i baz publikacji technicznych zabraniają używania ich treści w systemach sztucznej inteligencji. Z takich norm bierzemy więc tylko wzory i liczby, które trafiają do solvera, a ich tekst nie trafia ani do bazy wiedzy, ani do uczenia. Podobnie bywa z dokumentacją projektową w firmach. Stąd zamknięta infrastruktura, opisana w rozdziale 3.

Liczby muszą się zgadzać co do cyfry

W obliczeniach inżynierskich wynik prawie dobry jest wynikiem złym. Dlatego u nas rachunek robi solver: program, który dla tych samych danych zawsze zwraca ten sam wynik. Każda komenda naszego nowego silnika solverów przeszła bramkę: odtworzyła przykład ze źródła, czyli z podręcznika, normy albo materiałów szkoleniowych, z dokładnością do zaokrągleń tego źródła.

Sprawdzanie solverów przyniosło niespodziankę. Przy odtwarzaniu przykładów znaleźliśmy kilkadziesiąt rozbieżności. Część to pomyłki rachunkowe w samych książkach. W jednym przykładzie wartość pomnożono przez współczynnik wyporu, który uwzględnia, że rury zanurzone w cieczy ważą mniej, choć należało przez niego podzielić. W innym, w rachunku kosztu świdra, pominięto marsz, czyli czas wyciągania i zapuszczania przewodu przy wymianie świdra. Inne rozbieżności to literówki w stałych, błędy odczytu skanów i zaokrąglenia, które w zadaniach szkoleniowych z kontroli ciśnienia w otworze przesądzały o wyniku. Solver sprawdzony na kilku źródłach bywa więc pewniejszy niż pojedyncza książka.

> **Zasada.** Liczby liczy solver, wiedza pochodzi z dokumentów, a w wagach zostaje zachowanie.



Trzy miejsca na trzy rodzaje wiedzy. Dokumenty i solver poprawia się bez treningu modelu.

Każda specjalność ma swój język

Inżynier rozpoznaje fachowca po słowach. Narzędzie, które w otworze zwierca skałę, to w polskim wiertnictwie świder. Graniatka to graniasta rura, przez którą urządzenie obraca ciąg rur ze świdrem na końcu, i w polskim tekście nie powinna się nazywać angielskim słowem kelly. Każda specjalność ma takie pary słów, swoje i zapożyczone. Model ogólny tych rozróżnień nie musi znać.

Zbudowaliśmy na to test z 86 par zdań. Zdania w parze różnią się jednym terminem: jedno ma termin poprawny, drugie kalkę albo błąd. Model dostaje punkt, gdy poprawnemu zdaniu przypisuje wyższe prawdopodobieństwo, więc model zgadujący uzyskałby około 0,5.

MODEL	WYNIK W TEŚCIE TERMINOLOGII
Stanisław 1B	0,907
Bielik 11B, wersja Instruct	0,872
Goldfish 125M	0,779
Qwen3.8-27B	0,733

Model o 27 mld wag, ponad dwadzieścia razy większy od naszego, wypadł w tym teście najsłabiej. Nasz 1B wyprzedził też Bielika 11B w wersji Instruct; wersji bazowej w tym teście nie mierzyliśmy. Przy 86 parach niepewność wyniku wynosi jednak około 6-11 punktów procentowych, więc tej przewagi nie uznajemy za rozstrzygniętą. Wynik pasuje do tego, że model widział w pretreningu teksty z dziedziny, ale sam test tego nie dowodzi.

Co zostaje w wagach

Skoro rachunek robi solver, a fakty leżą w dokumentach, model nie musi być wielki. Ma umieć sięgnąć po właściwe narzędzie i przepisać do niego liczby z pytania. Ma wyczuć, czy wynik ma sensowny rząd wielkości, i zapytać o daną, której brakuje. Gdy narzędzia albo dokumentu nie ma, ma odmówić zamiast zgadywać.

Pomiary na modelu 1B to potwierdzają. Po strojeniu, czyli douczeniu na przykładach zadań z narzędziami, wybiera właściwe z pięciu narzędzi za każdym razem i w 96-100% przypadków dostaje poprawny wynik solvera, zależnie od zestawu testowego. Model ogólny o 27 mld wag, wyposażony w te same narzędzia, dał poprawny wynik w 99,6% pytań jednego z zestawów testowych. W wywoływaniu solvera mały model jest więc prawie na poziomie dużego, choć oba liczono na nie całkiem tych samych zestawach.

Wybór spośród pięciu narzędzi jest jednak łatwy. Prosty program, który szuka w pytaniu określonych słów i wzorców, bez żadnego uczenia, trafiał narzędzie w 93,8% pytań. Poprawny wynik dawał już tylko w 81%, a trafnie odmawiał w 51,5%. Prawdziwa próba zaczyna się przy 147 narzędziach, opisanych w rozdziale 4.

Trudniej jest powiedzieć „nie wiem” przy odpowiadaniu z dokumentów. Cztery kolejne strojenia 1B leżą na jednej krzywej: im częściej model trafnie odmawia w trudnych przypadkach, tym częściej odmawia też tam, gdzie odpowiedź w dokumentach jest. Tę granicę opisujemy w rozdziale 6.

Ten sam model 1B w testach wiedzy ogólnej z wyborem odpowiedzi wypada na poziomie losowym. To zgodne z zasadą: wiedza ma przychodzić z dokumentów, a liczby z solvera.

Czego własny model nie daje

Model od zera nie jest lepszy we wszystkim. Jakość modelowania tekstu mierzymy w bitach na bajt (bpb): to liczba bitów, jakiej model potrzebuje średnio na zapisanie jednego bajtu tekstu, którego nie widział, więc mniej znaczy lepiej. Bazowy Bielik 11B wypada w tej mierze wyraźnie lepiej od naszego 1B, o około 0,10 bpb na tekstach konkursu PolEval. Mniejszy Bielik 1.5B w wersji Instruct uzyskał w teście Belebele 66%, a nasz 1B w tym samym formacie odpowiada na poziomie zgadywania.

Spodziewamy się, że model od zera na samym polskim korpusie nie dogoni Bielika 11B w wiedzy ogólnej, także przy 7-11 mld wag; o różnicy w pochodzeniu obu modeli piszemy w rozdziale 2. Własny model ma sens tam, gdzie liczą się polszczyzna, terminologia, dziedzina i zachowanie: korzystanie z narzędzi, odpowiedź z dokumentu, odmowa. Do tego dochodzi kontrola nad danymi. Kiedy wystarczy gotowy model z dokumentami i solverem, piszemy w rozdziale 10.

Ocena przez ludzi

Stanisław ma być przede wszystkim asystentem inżyniera przy realnym projekcie otworu wiertniczego. Zadania egzaminacyjne są drugim wyznacznikiem, a model będą sprawdzać naukowcy Wydziału. To zmienia kolejność prac. Pytanie egzaminacyjne ma odpowiedź w kluczu, a projekt często nie ma kompletu danych. Łańcuch obliczeń, który przy brakującej danej zwraca listę braków zamiast przyjąć wartość po cichu, waży więc więcej niż kolejny pojedynczy kalkulator.

Na pytanie „kim jesteś” Stanisław ma odpowiadać, że jest pierwszym polskim modelem inżynierskim zintegrowanym z solverami. W chwili pisania jeszcze tego nie umie. W teście dymnym przed najbliższym strojeniem nie odpowiedział poprawnie na żadne z 14 pytań o tożsamość, bo zestaw uczący tożsamości wchodzi dopiero do tego strojenia.

ROZDZIAŁ 2

Od zera: co to znaczy i ile kosztuje

Co znaczy „od zera”

Od zera znaczy u nas trzy rzeczy. Korpus złożyliśmy sami z publicznych zbiorów tekstów, artykułów z czasopism i książek branżowych, a deduplikację między źródłami i odkażanie względem testów zrobiliśmy własnym kodem. Tokenizer wytrenowaliśmy na własnej próbce tego korpusu. Wagi modelu na początku pretreningu miały wartości losowe, więc wszystko, co dziś w nich jest, pochodzi z naszych danych.

Inna droga polega na dalszym uczeniu cudzego modelu. Tak, według opisu jego autorów, powstał Bielik 11B: dotrenowano go na polskim tekście, zaczynając od otwartego modelu Mistral. Taki model startuje z wiedzą z cudzego pretreningu, a o danych tej pierwszej części treningu decydował ktoś inny.

Architektura Stanisława nie jest nowa. To dekoder typu znanego z otwartych modeli Llama, a trener mieści się w jednym pliku wzorowanym na małych otwartych implementacjach. Nowe są dane i wagi.

Od zera nie znaczy też bez pomocy innych modeli. Część danych do strojenia, czyli parafrazy pytań oraz pytania i odpowiedzi do nauki pracy z dokumentami, wygenerowały otwarte modele uruchomione u nas jako nauczyciele. Pretrening nie korzysta z żadnego nauczyciela, a dane obliczeniowe dla solverów i odmowy syntetyczne generuje czysty kod, bez modelu językowego.

Drabina rozmiarów

Zanim uruchomiliśmy bieg na 41 godzin, sprawdziliśmy całą ścieżkę na modelach mniejszych o kilka rzędów wielkości. Jakość mierzymy w bitach na bajt (bpb). To liczba bitów, jakiej model potrzebuje średnio na zapisanie jednego bajtu tekstu, którego nie widział. Mniej znaczy lepiej, a miara nie zależy od tokenizera, więc pozwala porównać modele, które tną tekst inaczej.

Najmniejszy model, około miliona wag, służy za test dymny. Przez 3000 kroków, w około 4 minuty, czyta ok. 98 mln tokenów, a walidacja spada z 2,04 do 1,60 bpb. Jeśli ten test przechodzi, środowisko jest sprawne.

Model 30 mln wag uczyliśmy w trzech kształtach, od szerokiego i płytkiego po wąski i głęboki, na ok. 2,08 mld tokenów, w 4,1-5,5 h na karcie Tesla V100. Kształty różniły się o ok. 0,01 bpb, więc przy tej skali nie mają znaczenia. Model 125 mln wag był o 0,12 bpb

lepszy od 30M. Pokazał też, że tekst z sieci jest w mieszance potrzebny: wersja uczona tylko na kuratorowanym korpusie DynaWord przegrywała na tekstach webowych.

	STANISŁAW 1B	STANISŁAW 3B
Wagi	1,198 mld	3,045 mld
Warstwy i szerokość	22 warstwy po 2048	26 warstw po 3072
Kontekst	2048 tokenów	2048 tokenów
Tokeny pretreningu	ok. 28 mld	60 mld
Kroki	26 700	57 000
Tempo na 8 kartach	ok. 200 tys. tokenów/s	ok. 77,6 tys. tokenów/s
Czas	41 h	ok. 9 dni, bieg trwa
Punkt kontrolny ze stanem optymalizatora	ok. 14,4 GB	36,5 GB

> **Zasada.** Najpierw tani test całej ścieżki na małym modelu, dopiero potem duży bieg.

Korpus i jego czyszczenie

Podstawą korpusu jest polska część otwartego zbioru stron WWW FineWeb-2, uzupełniona rankingiem jakości, który wskazuje górne 10% dokumentów. Dochodzą teksty z plików PDF (nauka, normy, technika), archiwum stron HPLT, kuratorowany korpus DynaWord, 2340 artykułów z polskich czasopism branżowych i książki branżowe. Przed deduplikacją było to 1175 plików; same największe zbiory zajmowały kilkaset gigabajtów, a korpus po deduplikacji zajmuje 243 GB.

Te same teksty krążą po wielu zbiorach, więc pierwszym zadaniem była deduplikacja między źródłami. Program porównuje dokumenty trzema sposobami: po skrócie całego tekstu, po adresie strony i metodą MinHash. Ta ostatnia buduje z pięciowyrazowych fragmentów podpis dokumentu i szybko znajduje teksty prawie identyczne. Z każdej grupy duplikatów zostaje kopia z najcenniejszego źródła. Na 64 rdzeniach procesora trwało to ok. 2,3 h. Liczba dokumentów spadła z 213,6 do 179,0 mln, a liczba znaków z 925 do 715 mld, czyli o 22,7%.

Wynik sprawdziliśmy na losowej próbie 20 usuniętych par. Osiemnaście było prawdziwymi duplikatami, ale dwie miały ten sam adres i inną treść. Reguła adresu okazała się za ostra i odebrała korpusowi ok. 1,62 mln dokumentów, które trzeba przywrócić. Te 715 mld znaków to ok. 161 mld tokenów obecnego tokenizera.

Drugie czyszczenie, odkażanie, chroni pomiar. Jeśli tekst z zestawu testowego trafi do treningu, model wypadnie w teście lepiej, niż naprawdę umie. Dlatego usuwamy z danych treningowych każdy dokument, który dzieli z tekstem testowym co najmniej 10 różnych ciągów po 13 tokenów, co odpowiada mniej więcej 22 kolejnym wspólnym tokenom. Przy krótkich tekstach testowych próg jest odpowiednio niższy. Pierwsza kontrola przed startem 3B pokazała, że skażonych było 15% dokumentów testu PolEval. Po drugim przejściu skażenie spadło do 0% dla tekstów z naszej dziedziny i do 1,1% dla PolEvalu, przy progu 1%. Uznaliśmy dane za czyste, a wyniki PolEvalu podajemy na czystym podzbiorze.

Osobno odłożyliśmy 137 artykułów z czasopism, których model nie widzi w treningu. Z oczyszczonego korpusu powstają mieszanki, czyli pliki z zadanym udziałem tokenów z każdego źródła. Mieszanka 3B przed odkażaniem miała 60,4 mld tokenów treningu, 6,2 mld na końcową fazę wyżarzania i 0,17 mld do walidacji, a najwięcej, 22,1 mld, dawał w niej FineWeb-2. Po odkażaniu bieg 3B dostał 55,3 mld tokenów treningu i 5,3 mld do wyżarzania. Polskie książki branżowe to tylko ok. 13 mln unikalnych tokenów, ale w treningu powtarzają się 11,5 raza.

Nie wszystko się udało. Wersja Pythona na serwerze treningowym uruchamia procesy pomocnicze inaczej niż starsze wersje i część filtrów po cichu nie działała. W mieszance 1B najpewniej część dokumentów trafiła przez to do treningu dwa razy; zostawiliśmy to do sprawdzenia, bez naprawy wstecz. W książkach branżowych tabele zapisano jako surowy kod HTML, który w próbie polskich podręczników stanowi ok. 32% znaków. Przeszedł do mieszanki 3B razem z tekstem i przed następnym modelem trzeba go oczyścić.

Tokenizer

Model nie widzi liter, tylko tokeny, czyli kawałki tekstu ze stałego słownika. Dobry tokenizer polski zostawia częste terminy w całości, na przykład „ciśnienie", „hydrostatyczne" i „świder". Tekst zajmuje wtedy mniej tokenów, a model widzi więcej treści w tym samym oknie.

Nasz tokenizer to SentencePiece w odmianie Unigram ze słownikiem 32 768 kawałków. Uczyliśmy go na próbce ok. 2 mld znaków z dziesięciu źródeł, w której książki branżowe miały sześciokrotną wagę. W porównaniu z poprzednią wersją mieści więcej znaków w jednym tokenie: 4,78 zamiast 4,06 na najlepszych tekstach z sieci i 4,00 zamiast 3,59 na tekstach z PDF.

Słownik ma 69 tokenów specjalnych. Trzynaście opisuje strukturę rozmowy, na przykład `<pytanie>`, `<wywolanie>`, `<wynik>` i `<odmowa>`. Sześć nazywa narzędzia, a 50 czeka w rezerwie. Cyfry tokenizer zawsze tnie pojedynczo.

Tokenizer ma jedną poważną wadę, którą znaleźliśmy dopiero przy planowaniu następnego modelu. Jego normalizacja zamienia znak nowej linii na spację. Modele 1B i 3B nie widziały więc akapitów, list ani tabel. Dlatego tabele i listy w odpowiedziach ma składać program wokół modelu, a następny tokenizer zachowa nowe linie. Angielski jest w obecnym tokenizerze wyraźnie droższy: w angielskich tekstach naukowych z PDF to 2,13 tokenu na słowo, w polskich tekstach z sieci 1,57.

Pretrening 1B i 3B

Pretrening polega na tym, że model czyta tekst i w każdym miejscu przewiduje następny token. Strata mierzy, jak bardzo się myli. Krok to jedna poprawka wag po przeczytaniu porcji tekstu; w obu naszych biegach porcja ma ok. 1,05 mln tokenów. Co kilkaset kroków trener zapisuje punkt kontrolny: wagi oraz stan optymalizatora, czyli pamięć programu, który poprawia wagi, o tym, jak je dotąd poprawiał. Z takiego zapisu da się wznowić bieg.

Szybkość uczenia rośnie przez pierwsze 2000 kroków, w fazie rozgrzewki, potem jest stała, a w ostatnich 10% kroków liniowo maleje do jednej dziesiątej. Tę ostatnią fazę nazywamy wyżarzaniem i podajemy w niej teksty lepszej jakości: prozę fachową z PDF, Wikipedię, artykuły naukowe i literaturę.

Model 1B przeczytał ok. 28 mld tokenów w 26 700 krokach, w 41 h na ośmiu kartach Blackwell. W trakcie biegu dołożyliśmy do wyżarzania 2226 artykułów branżowych, każdy trzykrotnie. Wymagało to zatrzymania treningu, podmiany pliku danych i wznowienia z punktu kontrolnego; przerwa trwała 30 sekund. Dla modelu 1B w wersji po strojeniu SFT6 źródła podają 0,583 bpb na walidacji mieszanki, 0,676 bpb na wszystkich 137 odłożonych artykułach z dziedziny i 0,907 w teście terminologii.

Potem rozszerzyliśmy kontekst 1B, czyli długość tekstu widzianego naraz, z 2048 do 8192 tokenów dalszym pretreningiem na ok. 1 mld tokenów z przewagą długich dokumentów, w ok. 7 h na dwóch kartach. Sprawdza to test „hasło w stogu siana”: pięciocyfrowe hasło ukryte w długim dokumencie. Model przed rozszerzeniem znajdował je zawsze w tekstach o długości 2048 tokenów, a w dłuższych ani razu. Po rozszerzeniu trafia w 97,5% przypadków przy 4 tys. i 8 tys. tokenów, a w tekstach o długości 2048 tokenów nadal zawsze.

Model 3B uczy się na 60 mld tokenów w 57 000 kroków. Na dwóch kartach zmierzyliśmy 19,5 tys. tokenów na sekundę, na czterech 35,6 tys., na ośmiu ok. 77,6 tys. Bieg ruszył 10 listopada i ma się skończyć ok. 19 listopada. Walidacja spadała z 1,257 bpb w kroku 1000 do 0,844 w kroku 6000. Tych liczb nie porównujemy wprost z krzywą 1B, bo zbiory walidacyjne obu modeli są różne.

Co pretrening daje, a czego nie

Model bazowy 1B dobrze zna polską gramatykę. W teście gramatyki MultiBLiMP uzyskał 0,984, prawie tyle co Bielik 11B w wersji Instruct (0,989). Wiedzy o świecie prawie nie ma. W polskich wersjach testów HellaSwag i ARC-Easy ma 0,435 i 0,443 wobec 0,706 i 0,684 tej samej wersji Bielika. W testach wiedzy z wyborem odpowiedzi jest na poziomie losowym. W Belebele z wyborem litery A-D ma 23,3% przy 25% dla zgadywania, a sygnał pojawia się dopiero, gdy ocenia się tekst odpowiedzi (33,1%).

To zgodne z założeniem projektu: model 1B ma być sprawnym modelem języka, a wiedzę dostaje z dokumentów, liczby z solvera. Po 3B spodziewamy się, że testy wiedzy z wyborem odpowiedzi wyjdą ponad poziom losowy. Powinien też wzrosnąć odsetek trafnych odmów w trudnych przypadkach pracy z dokumentami, na którym 1B zatrzymał się na ok. 0,65.

Zestawienie z Bielikiem

Bielik jest dla nas punktem odniesienia metody. Ma wersję bazową i wersję Instruct, dostrojoną do wykonywania poleceń. Naszą uprząż oceny sprawdziliśmy na bazowym Bieliku-11B-v2: odtwarza jego opublikowane wyniki z dokładnością do 1,2 punktu procentowego. W modelowaniu języka porównujemy bity na bajt na dwóch zbiorach: tekstach konkursu PolEval i odłożonych artykułach z naszej dziedziny. Na PolEvalu liczymy tylko 1082 dokumenty, które nie dzielą z danymi Stanisława żadnego ciągu 13 tokenów. Wśród artykułów z dziedziny zupełnie czystych nie było, więc bierzemy 98 takich, w których wspólnych ciągów jest mniej niż 10%.

TEST	STANISŁAW 1B	BIELIK 11B BAZOWY	BIELIK 11B INSTRUCT
PolEval, bpb, mniej znaczy lepiej	0,929 (kontekst 2k), 0,924 (8k)	0,830	0,940
Artykuły z dziedziny, bpb	0,723 (2k), 0,686 (8k)	0,616	0,679
Terminologia, 86 par, więcej znaczy lepiej	0,907	nie mierzono	0,872

Bazowy Bielik jest lepszy od 1B o ok. 0,10 bpb na PolEvalu i o 0,07-0,11 bpb na tekstach z dziedziny. Wcześniejsza wersja naszej strony porównywała 1B tylko z wersją Instruct i pokazywała przewagę 1B; poprawiliśmy to 11 listopada. Strojenie do poleceń najpewniej pogarsza bpb, więc poprzeczką dla 3B jest Bielik bazowy. Jedno zastrzeżenie działa w obie strony: czystość podzbioru liczymy względem danych Stanisława i nie wiemy, czy Bielik nie widział tych tekstów.

Spodziewamy się, że model od zera na samym polskim korpusie nie dogoni Bielika 11B w wiedzy ogólnej nawet przy 7-11 mld wag. Porównujemy się w polszczyźnie, terminologii, dziedzinie i zachowaniu.

Prognoza skalowania i koszt

Dla modeli większych niż 3B mamy szacunki, nie pomiary. Czas liczymy z reguły dopasowanej do biegów 1B i 3B: tempo na kartę wynosi ok. 28/N tys. tokenów na sekundę, gdzie N to liczba miliardów wag. Dla 5B i więcej doliczamy ok. 30% na komunikację między kartami. Zysk szacujemy prawem skalowania, które opisuje, jak strata spada z liczbą wag i tokenów; przyjmujemy w nim 20 tokenów na każdą wagę, jak w znanej pracy o modelu Chinchilla. To prawo dopasowano do korpusów angielskich, więc dla polszczyzny i naszego tokenizera wskazuje kierunek i rząd wielkości, a nie prognozę punktową.

Ostatnia kolumna tabeli mówi, o ile maleje redukowalna część straty, czyli ta, którą da się jeszcze zmniejszyć większym modelem i większą ilością danych.

MODEL	TOKENY	CZAS NA 8 KARTACH	REDUKOWALNA STRATA WOBEC 1B
1B	28 mld	41 h, pomiar	100%
3B	60 mld	ok. 9 dni, tempo zmierzone	o 23% mniej
5B	100 mld	ok. 4 tygodnie	o 33% mniej
7B	140 mld	ok. 2,5 miesiąca	o 40% mniej
9B	180 mld	ok. 4 miesiące	o 44% mniej
11B	220 mld	ok. 6 miesięcy	o 48% mniej

Krzywa zysku się spłaszcza, a czas rośnie liniowo z rozmiarem i liczbą tokenów. Przejście z 1B do 3B daje 23% za ok. pięć razy więcej czasu. Przejście z 3B do 7B daje kolejne 17 punktów za ok. siedem razy więcej.

Pierwszą barierą jest pamięć karty. Obecny trener trzyma na każdej karcie pełne wagi, gradienty i stan optymalizatora, ok. 16 bajtów na wagę. Dla 5B to 80 GB, zanim policzy się cokolwiek innego, przy karcie 96 GB. Większe modele wymagają podziału wag i stanów między karty; nowy trener z takim podziałem sprawdziliśmy na małym modelu.

Drugą barierą są dane. Model 11B według reguły 20 tokenów na wagę potrzebuje 220 mld tokenów, więcej, niż ma nasz polski korpus, więc trzeba by powtarzać teksty. Według badań nad powtarzaniem danych do ok. czterech epok prawie nie traci się jakości. Dla 7B na 100 mld tokenów polskiego tekstu starcza z zapasem. Wąskim gardłem jest tekst z dziedziny: polskie podręczniki to ok. 13 mln tokenów.

Kosztu w pieniądzach jeszcze nie policzyliśmy. W czasie maszynowym koszt jest jasny: 1B to niecałe dwie doby ośmiu kart, 3B ok. dziewięciu dni. Model 7B na 100 mld tokenów według dokładniejszego planu zająłby tych samych kart ok. 49 dni. Dlatego najpierw kończymy 3B i oceniamy go na czystych testach, a o większym modelu zdecydujemy po tej ocenie. Limitem są dane i czas, bardziej niż sam rozmiar modelu.

ROZDZIAŁ 3

Zamknięta infrastruktura

Co znaczy „zamknięta”

Zamknięta znaczy, że pretrening, strojenie, wagi, modele-nauczyciele i praca Stanisława odbywają się na serwerach, z których korzystamy bezpośrednio, i żadna z tych rzeczy nie korzysta z usług zewnętrznych. Tam, gdzie do przygotowania danych potrzebny jest inny model językowy, na przykład do parafrazowania pytań, uruchamiamy otwarte wagi na kartach tych serwerów i odpytujemy je lokalnie. Solvery działają na procesorach serwera wydziałowego. Pokaz pracy modelu z solverem działa tylko lokalnie, bez wystawiania na świat.

Inaczej jest z warsztatem. Kod, przeglądy danych, katalog wzorów i analizę wyników przygotowywaliśmy z pomocą asystentów AI działających jako usługi zewnętrzne. Asystenci ci widzieli fragmenty danych, testów, odpowiedzi modelu i podręczników. Wagi Stanisława nie powstały na ich serwerach, ale ta część pracy zamknięta nie była.

Publicznie pokazujemy jedną rzecz: stan biegu. Co dwie minuty numer kroku, tempo, strata, ostatnia walidacja, alarmy czujki i obciążenie kart trafiają na stronę projektu. Treści korpusu ani rozmów tam nie ma.

Zamknięta nie znaczy, że wszystko jest nasze. Serwer z ośmioma kartami Blackwell, na którym trwa pretrening, należy do zaprzyjaźnionego operatora. Przez część prac połowa jego kart była zajęta innymi zadaniami właściciela. Bieg dłuższy niż miesiąc to ryzyko przerw niezależnych od nas. Wznawianie z punktu kontrolnego mamy gotowe, ale dostępność maszyny trzeba uzgadniać z jej właścicielem.

Gotowy model jest mały. Wagi 1B zajmują 2,4 GB w formacie bf16, jednym z 16-bitowych zapisów liczb używanych przy uczeniu. Model 3B, a według naszego szacunku także przyszły 7B, mieści się w drugim takim zapisie, fp16, na jednej starszej karcie Tesla V100.

Sprzęt

Pracujemy na dwóch rodzajach kart graficznych. Pretrening idzie na ośmiu kartach RTX PRO 6000 Blackwell po 96 GB pamięci. Łączy je magistrala PCIe, bez szybkiego łącza NVLink między kartami. Cztery z nich mają limit mocy 475 W, a cztery to wersje 300 W. Przy uczeniu rozproszonym wszystkie karty czekają na najwolniejszą, więc słabsza połowa wyznacza tempo całości.

Ewaluacja, strojenie modelu 1B, solvery i generowanie parafraz idą na kartach Tesla V100 serwera wydziałowego. Kart V100 o pamięci 32 GB mamy w zasięgu czternaście: sześć na serwerze wydziałowym i osiem na drugim serwerze. To starsze karty, które liczą w formacie fp16, a nie bf16 jak Blackwelle. Na serwerze wydziałowym Stanisław ma dwie karty. Pozostałe obsługują inne usługi Wydziału i ich nie dotykamy.

Między serwerami jest wolne łącze, ok. 0,6 MB/s. Przeniesienie jednego pliku wag 1B trwało ok. godziny. Dlatego ocenę narzędzi robimy dwuetapowo: odpowiedzi modelu generujemy tam, gdzie leżą wagi, a solverem sprawdzamy je tam, gdzie działa solver. Przenosi się tylko mały plik z tekstem odpowiedzi.

Osobnym ograniczeniem jest dysk. Jeden punkt kontrolny 3B ze stanem optymalizatora zajmuje 36,5 GB i zapisujemy go co 200 kroków. Dysk roboczy serwera bywał zajęty w 93-96%.

Jak pilnujemy wielodniowego biegu

Bieg 3B trwa ok. dziewięciu dni i nikt nie siedzi przy nim cały czas. Pilnuje go czujka, czyli osobny program, który co minutę sprawdza, czy żyją wszystkie procesy treningu i czy dziennik biegu jest świeży. Sprawdza też, czy strata jest skończona i nie skoczyła o więcej niż 1,0 ponad minimum ostatniego okna, a walidacja nie wzrosła o więcej niż 0,15.

Po każdym punkcie kontrolnym, za którym dziennik pokazał skończoną stratę, czujka oznacza go jako sprawdzony. Robi to twardym dowiązaniem, czyli drugą nazwą tego samego pliku, bez kopiowania 36 GB. Gdy procesy treningu padną, czujka sama wznawia bieg z ostatniego punktu kontrolnego, najwyżej trzy razy. Gdy strata wygląda źle, nie wznawia niczego i zostawia decyzję człowiekowi.

Trener ma własny bezpiecznik: zatrzymuje się, gdy strata stanie się nieskończona albo przestanie być liczbą. Działa on jednak dopiero po rozgrzewce, więc przez pierwsze 2000 kroków bieg chroni tylko czujka. Odnotowaliśmy to jako lukę.

Walidacja liczona na losowej próbce bywa kapryśna. W kroku 7000 walidacja 3B podskoczyła, a w kroku 8000 znów spadła; skok był szumem. Od tamtej pory pojedynczy wzrost walidacji przy spadającej stracie treningowej czujka zgłasza jako uwagę, a nie alarm.

Czujka ma też zadanie na przyszłość. Zachowuje punkt kontrolny z kroku 51 200, tuż przed wyżarzaniem. Z niego ruszy drugi wariant końcówki 3B, z 30% tekstu angielskiego, który porównamy z wariantem tylko polskim na tych samych testach.

Czego nauczyły nas awarie

W nocy z 11 na 12 listopada jedna z kart Blackwell wypadła z magistrali i bieg 3B stanął na kroku 9600. Ostatni punkt kontrolny i ostatni sprawdzony punkt były całe. Próba programowego odłączenia uszkodzonej karty zawiesiła się w jądrze systemu i od tej chwili żaden nowy proces nie mógł użyć żadnej karty. Nie wiemy, czy karty działały jeszcze przed tą próbą. Programowy restart nie dokończył się sam i serwer wstał dopiero po restarcie fizycznym. Potem wszystkie osiem kart przeszło test i bieg ruszył od kroku 9600. Po niecałej godzinie karta znów zawiodła; właściciel wymienił jej zasilanie i bieg wznowił się od kroku 9800.

Przy takiej awarii tracimy kroki od ostatniego punktu kontrolnego. Przy zapisie co 200 kroków i tempie ok. 13,5 s na krok to najwyżej ok. 45 minut biegu.

Tej samej nocy zanik zasilania przerwał strojenie modelu 1B na kroku 950. Strojenie wznowiliśmy z zapisanego kroku 500, więc 450 kroków trzeba było powtórzyć.

Najwięcej nauczyły nas awarie ciche. Ocena jednego ze strojeń uruchomiła się ze starym tokenizerem o słowniku 16 tys. kawałków zamiast obecnego. Model generował bezsensowny tekst, a próba wyglądała na udaną, bo sprawdzaliśmy kod wyjścia programu, a nie treść. Dziś skrypt oceny odmawia startu, gdy słownik tokenizera nie zgadza się z modelem, a każdą próbę porównujemy z odpowiedziami znanymi z wcześniejszego biegu. W innym przypadku skrypt bez zabezpieczenia na błąd wpisał znacznik sukcesu po tym, jak trening padł. Każdy krytyczny krok kończy się teraz jawnym znacznikiem błędu, a kolejne etapy go czytają.

Pamięć zawodziła kilka razy. Najgroźniej, gdy program tokenizujący trzymał cały korpus w pamięci i uruchomił 80 procesów: każdy skopiował ten sam korpus, serwer przestał odpowiadać na ok. 10 minut, a system zabił proces. Osobną lekcją był model nowej architektury uruchomiony przez ogólną bibliotekę na kartach V100. Po błędzie jąder obliczeniowych żadna karta serwera nie przyjmowała zadań aż do restartu.

Z tych zdarzeń zostały zasady, które stosujemy przy każdym biegu:

- punkt kontrolny zapisujemy często, przez plik tymczasowy podmieniany na końcu zapisu, i sprawdzamy, że da się z niego wznowić;
- sprawdzamy treść wyniku, a nie kod wyjścia programu;
- ciężkie zadania na procesorze uruchamiamy z twardym limitem pamięci i ograniczoną liczbą procesów;
- model nowej architektury uruchamiamy tylko w zestawie kart i bibliotek, w którym go wcześniej sprawdzono;
- przed biegiem na kilka dni ustalamy z właścicielem maszyny, które karty są wolne.

Prawa do danych

Dla każdego źródła korpusu zapisujemy podstawę, na której z niego korzystamy. Zbiory stron WWW i plików PDF są udostępnione na licencji ODC-By, archiwum HPLT ma licencję CC0 na opakowanie zbioru, a korpus DynaWord ma licencję przypisaną do każdego dokumentu. Jego część z wykładów TED licencji nie ma, więc ją pominęliśmy. Artykuły z czasopism mają licencje CC BY albo CC BY-SA według wydawcy.

Przy zbieraniu artykułów nie obchodzimy blokad. Gdzie wydawca chroni stronę przed programami pobierającymi, sięgamy po teksty tylko za jego zgodą. Zapytania wysyłamy nie częściej niż ok. raz na sekundę, z adresem kontaktowym w nagłówku.

Książki branżowe są chronione prawem autorskim. Korzystamy z nich na podstawie art. 26^2 prawa autorskiego, który pozwala na eksplorację tekstów i danych do badań naukowych bez korzyści majątkowej i wymaga legalnego dostępu do źródła. Teksty leżą tylko na maszynach projektu i nie publikujemy ich, a z serwera operatora usuniemy je, gdy przestaną być potrzebne. Przy budowie katalogu wzorów ich fragmenty czytali asystenci AI, z których korzystaliśmy w pracy. Norm nie pobieramy z serwisów z nieoficjalnymi kopiami. Z norm, których licencje wykluczają użycie w systemach sztucznej inteligencji, bierzemy do solverów tylko wzory i liczby, a ich tekst nie trafia ani do bazy wiedzy, ani do uczenia.

ROZDZIAŁ 4

Solvery jako organy modelu

Organ, nie kalkulator obok

Solver to w naszym słowniku deterministyczny program obliczeniowy: dla tych samych danych zawsze daje ten sam wynik i niczego nie zgaduje. Nazywamy solvery organami modelu, bo Stanisław ma po nie sięgać odruchowo, tak jak inżynier sięga po tablice i kalkulator, zanim poda liczbę. Model rozpoznaje rodzaj obliczenia, przepisuje dane z pytania, woła solver i opisuje wynik. Sam nie liczy.

Mechanizm składa się z czterech części: katalogu obliczeń z bramką, silnika solverów, rejestru narzędzi i pętli, w której model z nich korzysta. Żadna z nich nie zależy od dziedziny. Katalog przyjmie wzór z wytrzymałości materiałów tak samo jak wzór z hydrauliki otworowej, a rejestr i pętla nie wiedzą, czego dotyczy komenda. Pierwszym wypełnieniem tej formy są obliczenia wiertnicze. Zaczynamy od technologii pozyskiwania węglowodorów, a w nich od wierceń, bo tam mamy podręczniki, zadania i ludzi, którzy sprawdzą wynik.

Katalog obliczeń i bramka

Katalog to spis obliczeń, które inżynier wykonuje przy projekcie i na egzaminie, zebrany z 67 podręczników i materiałów szkoleniowych. Każda pozycja ma wzór i zmienne z jednostkami, przykład ze źródła z kompletem danych i wynikiem, własne przeliczenie tego przykładu, odchyłkę i stan bramki. Dziś katalog obejmuje dziewięć dziedzin wierceń głębokich: osiem rachunkowych w tabeli i dziewiątą, jednostki, ze 115 przeliczeniami i 76 stałymi.

DZIEDZINA	OBLICZENIA	PO BRAMCE
Cementowanie i płuczki wiertnicze	42	37
Kontrola otworu	42	37
Przewód wiertniczy	41	37
Hydraulika	39	33
Urządzenie, świdry, koszty	35	32
Ciśnienia	32	28
Rury okładzinowe	32	27
Wiercenie kierunkowe	32	30
Razem	295	261

Dla czytelnika spoza branży trzy objaśnienia. Przewód wiertniczy to ciąg rur, na którym świder pracuje w otworze. Rury okładzinowe obudowują gotowy odcinek otworu. Kontrola otworu to panowanie nad ciśnieniem w otworze, tak żeby płyn ze złoża nie wdarł się do niego w niekontrolowany sposób.

Bramka ma jedną regułę. Wzór wchodzi do solvera tylko wtedy, gdy test odtwarza co do cyfry przykład z podręcznika, normy albo materiałów szkoleniowych, z tolerancją wynikającą z zaokrągleń źródła. Test ma w komentarzu źródło i numer przykładu. Wzór bez zielonego testu nie istnieje, a wzór bez źródła, wpisany z pamięci, nie wchodzi wcale. Pozycja, która nie przeszła, zostaje w katalogu z oznaczeniem: przykład się nie odtwarza, przykładu brak albo obliczenie wykracza poza nasze źródła.

Bramka zatrzymała na przykład wzory na zgniatanie rur okładzinowych pod ciśnieniem zewnętrznym. W pierwszej wersji katalogu część ich współczynników wpisano z pamięci, bez źródła, więc silnik ich nie przyjął. Później sprawdziliśmy solver na przedruku tabel z podręcznika: wszystkie pięć wyników tabelarycznych się zgodziło, a przy współczynnikach przepisanych dosłownie z przedruku cztery z pięciu. Rozbieżność dla jednego gatunku stali opisaliśmy jawnie. Opisy komend rozdzielają przy tym wydania historyczne norm od współczesnych.

Osobną dziedziną są jednostki. Podręczniki amerykańskie liczą w funtach na galon, stopach i psi, polska praktyka w atmosferach technicznych i kilogramach na litr, a solver w SI. Stałe polowe, jak 0,052 czy 2,448, wyprowadzamy w teście z definicji jednostek, zamiast przepisywać je z książki. Pułapek jest sporo: tona krótka i metryczna, galon amerykański, atmosfera techniczna i fizyczna, kilogram-siła i kilogram, funt masy i funt siły. Te same pułapki czekają na każdego, kto przelicza dane z amerykańskiej dokumentacji, niezależnie od branży.

Silnik

Obliczenia z katalogu przenosimy do silnika napisanego w języku Rust. Ma dziś 142 komendy i przechodzi 168 testów. Jego program wywoływany z wiersza poleceń przechodzi 220 przypadków z poprawnymi danymi i 3424 przypadki z danymi błędnymi, które silnik musi odrzucić.

Kontrakt silnika jest ważniejszy niż liczba komend, bo od niego zależy współpraca z modelem. Komenda czyta jeden obiekt JSON i zwraca jeden obiekt JSON. Parametry mają jednostkę w nazwie, na przykład `rho_kg_m3` albo `h_m`, a wnętrze liczy w SI. Wynik zawiera liczby, kroki pośrednie z nazwami i źródło wzoru, żeby człowiek mógł sprawdzić tok obliczeń. Przy złych danych silnik kończy pracę błędem i komunikatem po polsku. Nigdy nie zwraca wartości nieliczbowej ani cichego wyniku poza zakresem ważności metody.

Komenda `lista` wypisuje wszystkie komendy z opisem, wejściami, zakresami i źródłem. Nazwy są krótkie, polskie i mają przedrostek dziedziny, na przykład `hydr_ecd` albo `kontr_karta_erupcji`. Taki kontrakt nie zawiera niczego, co wiązałoby go z wierceniem: równie dobrze opisze komendę liczącą ugięcie belki.

Kolejność budowy szła od podstaw. Najpierw jednostki i zadania z kolokwiów wydziałowych, potem hydraulika, kontrola otworu, ciśnienia, rury, przewód wiertniczy i cementowanie. Dalej arkusze projektu otworu, które łączą komendy w łańcuch, z listą brakujących danych i jawnymi założeniami. Arkusz otworu pionowego po zmianie geometrii albo płuczki przelicza wszystkie wyniki zależne. Brak danej zatrzymuje odpowiedni krok i trafia na listę braków, a arkusz niczego nie dopowiada. Na razie to przykład syntetyczny, sprawdzony 48 testami; pełnych projektów ze źródeł, które arkusz miałby odtworzyć, jeszcze nie mamy.

Rejestr narzędzi

Model nie woła silnika wprost. Między nimi stoi rejestr narzędzi: jedna lista, która łączy pięć narzędzi znanych modelowi 1B ze 142 komendami nowego silnika, razem 147 pozycji. Pięć starszych narzędzi liczy ciśnienie hydrostatyczne, stratę ciśnienia przepływu w rurze, dopasowanie modelu reologicznego do pomiarów, przeliczenie pomiarów kierunkowych otworu i geometrię trajektorii przewiertu sterowanego (HDD). Każda pozycja rejestru ma nazwę, jednozdaniowy opis po polsku, wejścia z jednostkami i zakresami oraz wyjścia. Rejestr buduje się z komendy `lista`, więc nowa komenda silnika od razu staje się narzędziem.

Osobnym problemem były jednostki w pytaniach. Inżynier pisze grubość ścianki w milimetrach, gęstość w kilogramach na decymetr sześcienny, wydatek w galonach na minutę. Gdyby model miał przeliczać jednostki sam, musiałby liczyć, a tego nie chcemy. Każde pole ma więc warianty z jednostką. Model przepisuje liczbę z pytania dokładnie

tak, jak stoi, i wybiera wariant pola z jednostką z pytania, na przykład `grubosc_mm` zamiast pola w metrach. Na SI przelicza ją rejestr komendami przeliczeń, które też przeszły bramkę, więc na jedno pytanie przypada jedno wywołanie.

W drugą stronę rejestr skraca wynik solvera do tego, co zobaczy model: najwyżej cztery wartości, w jednostkach, w których podaje je inżynier (MPa, kN, mm, l/s), z czterema cyframi znaczącymi.

Wspólna lista ma też pułapki. Dwie nowe komendy dublowały stare narzędzia do ciśnienia hydrostatycznego i do przeliczania pomiarów kierunkowych. Model uczony na ok. 60 tys. przykładów ze starymi nazwami dostałby dwie etykiety na to samo pytanie, a w teście dymnym już je mylił. Duble wyłączyliśmy z pilotażu, a nazwy ujednolicimy przed strojeniem 3B.

Rejestr jest już większy, niż zmieści kontekst modelu. Opisy wszystkich 147 narzędzi zajmują ok. 20 tys. tokenów, a nawet planowany dla 7B kontekst ma ich 4096. Przed pytaniem trzeba więc wybrać kilka do kilkunastu kandydatów, wyszukiwaniem po opisie albo małym modelem wyboru.

Pętla: pytanie, wywołanie, wynik, odpowiedź



Jedno pytanie obliczeniowe. Model wybiera narzędzie i przepisuje dane, liczbę daje solver.

Tak wygląda jeden rekord danych uczących z kandydata danych dla 3B. Pytanie, wywołanie, wynik i odpowiedź wygenerował kod z solvera. To wzorzec, którego model ma się nauczyć, a nie jego odpowiedź. Płuczka wiertnicza to ciecz, którą pompy tłoczą przez przewód do świdra i która wraca przestrzenią pierścieniową między przewodem a ścianą otworu. Komenda liczy równoważną gęstość cyrkulacyjną: gęstość płuczki w spoczynku, która dałaby na dnie to samo ciśnienie co płuczka krążąca, razem z oporami przepływu.

```
<pytanie>Strata ciśnienia w przestrzeni pierścieniowej wynosi 1 MPa. Gęstość
płuczki wiertniczej wynosi 1,15 kg/dm3, a głębokość pionowa TVD to 4000 m.
Potrzebuję wartości ECD do porównania z oknem gęstości.</pytanie>
<wywolanie><f:hydr_ecd>{"rho_kg_dm3": 1.15, "tvd_m": 4000.0,
"dp_pierscien_mpa": 1.0}</wywolanie>
<wynik>{"ecd_kg_m3": 1175.0, "przyrost_kg_m3": 25.49}</wynik>
<odpowiedz>Równoważna gęstość cyrkulacyjna ECD: 1175 kg/m³. Przyrost gęstości
równoważnej od oporów przepływu: 25,49 kg/m³.</odpowiedz>
```

Model w treningu uczy się pisać wszystko, co następuje po pytaniu, z wyjątkiem bloku `<wynik>`. Ten blok jest wyłączony z liczenia straty, bo w działaniu wstawia go solver. Model nie uczy się więc przewidywać liczb, tylko je przepisywać.

W działaniu pętlę prowadzi program wokół modelu. Model pisze wywołanie, a program pilnuje go na każdym tokenie. Pozwala wpisać tylko nazwę z rejestru, tylko pola z opisu narzędzia i tylko liczby, które padły w pytaniu. Po zamknięciu wywołania program woła solver i wstawia skrócony wynik, a model dopisuje odpowiedź. Gdy solver zwróci błąd, model dostaje go w bloku wyniku i ma go opisać. Pamięć klucz-wartość, czyli zapamiętywanie obliczeń dla już przeczytanego tekstu, przyspiesza całą pętlę 2,7-2,9 raza. Tekst odpowiedzi jest przy tym identyczny jak bez niej, w 30 z 30 pytań każdego z dwóch zestawów.

Pierwszą pełną pętlę z prawdziwym modelem sprawdziliśmy na zwykłym procesorze. Model 1B po strojeniu SFT8 dostał pytanie: „Policz ciśnienie hydrostatyczne płuczki o gęstości 1200 kg/m3 na głębokości 2000 m.” Wybrał narzędzie hydrostatyki, przepisał obie liczby, a solver zwrócił 23,54 MPa. Całość zajęła ok. 34 s. Zdanie odpowiedzi w tym pokazie złożył deterministyczny program, nie model. To była próba działania, nie pomiar skuteczności.

Że model nie powinien przepisywać długich tabel, pokazał pomiar na przeliczaniu pomiarów kierunkowych otworu metodą minimalnej krzywizny. Przy 13-16 stacjach pomiarowych, z których każda podaje głębokość, nachylenie i kierunek otworu, model, który sam przepisywał stacje do wywołania, trafiał w 0-5% przypadków. Dziś wpisuje tylko umowny znacznik, a stacje odczytuje z pytania deterministyczny parser; tak policzone wyszło 99 przypadków z 99. Gdy parser nie jest pewien odczytu, nie zgaduje, tylko oddaje pytanie człowiekowi.

Dopytanie, odmowa i metoda poza zakresem

Asystent przy projekcie częściej spotka pytanie bez kompletu danych niż pytanie z kluczem odpowiedzi. Dla braków mamy osobny znacznik `dopytaj`. Model zamiast wyniku wpisuje, czego brakuje, i zadaje jedno pytanie po polsku, z jednostkami, w których inżynier podaje tę wielkość. W pilotażowych danych uczących takie przykłady

to co najmniej 11% przykładów każdej komendy. Rekord z danych dla 3B, w którym brakuje gęstości płuczki potrzebnej do policzenia ciężaru elementu zanurzonego w płuczce:

```
<pytanie>Pomóż uzupełnić dane do tego zadania. Uwzględnij wypór cieczy: jaki
ciężar będzie przenoszony? Podano ciężar w powietrzu: 550 kN. Podano gęstość
materiału: 7.85 kg/dm3.</pytanie>
<wywolanie><f:dopytaj>{"narzedzie": "mech_wypor", "brakuje":
["rho_pluczki_kg_m3"]}</wywolanie>
<odpowiedz>Podasz gęstość płuczki wiertniczej (kg/dm³ lub kg/m³)?</odpowiedz>
```

Gdy pytanie nie wymaga obliczenia albo narzędzia nie ma, model wpisuje `<f:brak_narzedzia>` i znacznik `<odmowa>`, a potem odpowiada bez liczby. Trzeci przypadek jest najmniej oczywisty: narzędzie jest, ale dane leżą poza zakresem ważności metody. W jednym z rekordów pytanie dotyczy strat ciśnienia w przepływie burzliwym według korelacji Blasiusa, przy prędkości 0,015 m/s. Solver zwraca błąd: korelacja obejmuje tylko zakres 4000 < Re < 100 000. Wzorcowa odpowiedź w danych mówi, że metoda nie obowiązuje, bo liczba Reynoldsa leży poniżej tego zakresu, i nie podaje żadnej liczby.

Przy odmowach popełniliśmy błąd w pomiarze. W przeglądzie danych wyszło, że wszystkie 357 pytań z odmową w starym zestawie testowym narzędzi występowały też w danych uczących. Dawna miara odmów sprawdzała więc pamięć, a nie umiejętność. Dziś każda kopia testu ma pole oznaczające przeciek i patrzymy na wynik bez niego.

Model 1B po strojeniu SFT8 w teście dymnym daje poprawny wynik starymi narzędziami w 17 z 18 pytań, a nowych komend nie wybiera wcale, 0 z 20. Tego się spodziewaliśmy, bo ich nie widział. Uczy ich pilotażowe strojenie SFT11: 25 komend z różnych dziedzin, najczęstszych w pracy inżyniera i na egzaminach, w 12 500 przykładach, z dopytaniem i odmową.

Pilotaż skończył się 12 listopada. Na 802 pytaniach testowych, których nie było w danych uczących, model wybrał właściwe narzędzie w 96% przypadków, a solver wywołany z jego argumentami dał poprawny wynik w 94,8%. Wszystkie wywołania miały poprawny zapis. Przed pilotażem ten sam test dawał zero poprawnych wyników. Gdy brakowało danej, model trafnie dopytał w 86% przypadków, a gdy pytanie nie wymagało narzędzia, zawsze to rozpoznał. Stare narzędzia nie ucierpiały: na czterech dawnych zestawach wynik jest taki sam albo lepszy niż przed pilotażem.

Słabe miejsca też widać. Trzy komendy wypadły wyraźnie gorzej od reszty: objętość zaczynu w przestrzeni pierścieniowej oraz dwie komendy z karty likwidacji erupcji. W osobnym zestawie 36 pytań o komendy łatwe do pomylenia model wybrał właściwą w 25 przypadkach, a poprawny wynik dał tylko w 9. Te pary trzeba rozdzielić w danych, zanim przepis przejdzie na cały rejestr.

Dlaczego silniki są oddzielone od wag

Rozdzielenie solvera i wag ma praktyczny skutek. Gdy w silniku znajdziemy błąd albo dodamy wzór, poprawka działa od następnego wywołania, bez treningu. Model niczego o wzorze nie wie, zna tylko nazwę komendy i jej pola. Ten sam podział pozwala korzystać z norm, których tekstu nie wolno używać do uczenia: wzór i liczby trafiają do solvera, a tekst zostaje poza modelem.

Wymiana ma jedną granicę. Nazw komend i ich pól po rozpoczęciu strojenia nie zmieniamy, bo model nauczył się ich kształtu i dane uczące by się rozjechały. Każda zmiana wymaga zapisanej zmiany kontraktu, a model trzeba by douczyć na nowych nazwach. Nowe komendy mają działać od razu z opisu w kontekście, a do wag wejść w kolejnej rundzie. Zestaw ćwiczebny z 320 nazwami narzędzi, czyli 40 operacjami w ośmiu wariantach nazw, pól i jednostek, już jest, ale tej umiejętności modelu jeszcze nie zmierzyliśmy.

> **Zasada.** Solver poprawiamy w kodzie, a model uczymy tylko tego, kiedy i jak po niego sięgnąć.

W wagach zostaje to, czego żaden program nie zrobi za model: rozpoznanie, o jakie obliczenie chodzi, przepisanie danych i wyczucie, czy wynik ma sensowny rząd wielkości. Zostaje też decyzja, czy liczyć, dopytać, czy odmówić. Dla następnych modeli planujemy osobną miarę tego odruchu. Jedna jej część to odsetek pytań obliczeniowych, w których model liczy sam zamiast wołać solver, z celem równym zeru. Druga to liczby w odpowiedzi, których nie ma ani w wyniku, ani w pytaniu.

ROZDZIAŁ 5

Jak pracować ze Stanisławem

Pytanie, wywołanie, wynik, odpowiedź

Stanisław nie liczy w pamięci. Gdy pytanie wymaga rachunku, model pisze wywołanie narzędzia: nazwę komendy solvera i jej argumenty w zapisie JSON. Program wykonawczy przekazuje wywołanie do solvera, solver zwraca wynik, a model kończy odpowiedź zdaniem po polsku. Całą wymianę zapisujemy jako jeden tekst, w którym poszczególne części oddzielają tokeny specjalne `<pytanie>`, `<wywolanie>`, `<wynik>` i `<odpowiedz>`.

Nasze przykłady pochodzą z wierceń, bo tam zaczęliśmy, ale zasady pracy są takie same w każdej dziedzinie. Pierwszy przykład to próba na zwykłym procesorze, bez karty graficznej. Chodzi w nim o ciśnienie hydrostatyczne płuczki wiertniczej, czyli cieczy krążącej w otworze, której słup naciska na dno tak jak woda w zbiorniku. Model 1B po strojeniu SFT8 dostał polecenie: „Policz ciśnienie hydrostatyczne płuczki o gęstości 1200 kg/m3 na głębokości 2000 m.” Wygenerował takie wywołanie:

```
<f:hydrostatyka>{"rho_kg_m3": 1200, "h_m": 2000}
```

Solver zwrócił `{"p_MPa": 23.54}`. Odpowiedź brzmiała: „Ciśnienie hydrostatyczne na głębokości 2 000 m przy gęstości płuczki 1 200 kg/m3 wynosi 23,54 MPa.” W tej próbie zdanie odpowiedzi złożył z wyniku deterministyczny program, a nie model. Generowanie zajęło na procesorze 34 s.

Model zrobił tu dwie rzeczy: wybrał narzędzie i przepisał dane. Liczba 23,54 nie pochodzi z wag. Policzył ją program, który dla tych samych danych zawsze da ten sam wynik.

> **Zasada.** Model wybiera narzędzie i przepisuje dane z pytania. Liczy solver.

Jak pytać

Najlepiej pisać tak, jak pisze się zadanie dla kolegi z zespołu: wszystkie dane w treści, każda liczba z jednostką. Program wykonawczy pilnuje, żeby w argumentach wywołania stały wyłącznie liczby obecne w pytaniu. Model nie dopisze więc gęstości „z doświadczenia” ani nie zaokrągli głębokości. Czego nie ma w pytaniu, tego nie będzie w wywołaniu.

Jednostki mogą być takie, jakich używa się w praktyce, także spoza układu SI. Pytania w naszych danych strojenia celowo mieszają układy. Jest tam ciśnienie w psi, głębokość w metrach albo stopach, gęstość płuczki w kg/m3, kg/dm3 albo w funtach na galon, średnice w calach albo milimetrach. Każda komenda ma kilka wariantów pola z jednostką w nazwie. Model przepisuje liczbę dokładnie tak, jak stoi w pytaniu, i wybiera pole z pasującą jednostką. Przeliczenie na SI robi rejestr narzędzi, zanim solver zacznie liczyć.

Oto przykład z karty likwidacji erupcji, czyli arkusza obliczeń, którym opanowuje się niekontrolowany dopływ płynu ze skał do otworu. Pytanie podaje ciśnienie odczytane na przewodzie wiertniczym po zamknięciu otworu (SIDPP) 625 psi, głębokość pionową dna 2000 m i gęstość płuczki 1420 kg/m3. Wzorcowe wywołanie wygląda tak:

```
<f:kontr_cisnienie_zlozowe>{"rho_kg_m3": 1420.0, "tvd_m": 2000.0,
"sidpp_psi": 625.0}
```

Solver zwraca ciśnienie hydrostatyczne 27,85 MPa i ciśnienie złożowe, czyli ciśnienie płynu w skale, 32,16 MPa. Ten rekord pochodzi z przygotowanego zestawu danych do strojenia modelu 3B. To wzorzec, którego uczymy, a nie zapis rozmowy z modelem.

Liczby można pisać z przecinkiem albo z kropką, program traktuje oba zapisy jednakowo. Ułamków zwykłych rejestr jeszcze nie przyjmuje. Średnicę rury podajemy więc dziś jako 9,625 cala, a nie jako 9 5/8 cala, bo liczby 9,625 nie byłoby w pytaniu i model nie mógłby jej wpisać.

Jedno pytanie to jedno obliczenie. Ograniczenie, które każe przepisywać liczby z pytania, nie pozwala przepisać liczby z wyniku poprzedniego kroku. Dlatego w obecnych danych jest jedno wywołanie na pytanie. Ciąg obliczeń, w którym wynik jednej komendy jest daną dla następnej, przygotowujemy osobno jako arkusz projektu otworu wiertniczego.

Długie tabele danych są wyjątkiem od reguły przepisywania. W wierceniu kierunkowym mierzy się w kolejnych punktach, zwanych stacjami, długość otworu, jego nachylenie i kierunek, a metoda minimalnej krzywizny przelicza te odczyty na położenie otworu w przestrzeni. Model nie przepisuje kilkunastu trójek liczb. Pisze tylko, że stacje są w pytaniu, a odczytuje je deterministyczny parser. Gdy zapis jest niejednoznaczny, parser nie zgaduje i pyta człowieka. Ten wyjątek wziął się z pomiaru: przy 13-16 stacjach model, który sam przepisywał tabelę, trafiał w 0-5% przypadków, a z parserem w 99 na 99.

Gdy czegoś brakuje

Inżynier często pyta, zanim zbierze wszystkie dane. Stanisław ma wtedy dopytać, a nie zgadywać. W danych strojenia służy do tego pseudo-narzędzie `<f:dopytaj>`. Zamiast liczb model wypisuje nazwę komendy i listę brakujących pól, a w odpowiedzi zadaje jedno pytanie z jednostkami. Rekord z tego samego zestawu:

```
<pytanie>Pomóż uzupełnić dane do tego zadania. Uwzględnij wypór cieczy: jaki
ciężar będzie przenoszony? Podano ciężar w powietrzu: 550 kN. Podano gęstość
materiału: 7.85 kg/dm3.</pytanie>
<wywolanie><f:dopytaj>{"narzedzie": "mech_wypor", "brakuje":
["rho_pluczki_kg_m3"]}</wywolanie>
<odpowiedz>Podasz gęstość płuczki wiertniczej (kg/dm³ lub kg/m³)?</odpowiedz>
```

Chodzi o wypór: element zanurzony w cieczy waży mniej niż w powietrzu, a do policzenia ciężaru brakuje gęstości cieczy. Bloku `<wynik>` nie ma, bo solver nie dostał zadania. Pytanie zwrotne podaje jednostki, w których inżynier zwykle tę gęstość zna. Dane pilotażu, na których uczył się model 1B, dopytują jeszcze o wielkości w jednostkach SI, na przykład o ciśnienie w paskalach i średnicę w metrach. W kandydacie danych dla 3B poprawiliśmy to po przeglądzie próbki rekordów, bo w praktyce mało kto podaje ciśnienie w paskalach. W pilotażu strojenia co najmniej co dziewiąty przykład każdej komendy uczy dopytania.

Druga sytuacja to dane spoza zakresu ważności metody. Solver nie zwraca wtedy liczby, tylko komunikat błędu po polsku, a model ma powiedzieć, że metoda nie obowiązuje. W jednym z rekordów ktoś pyta o straty w gładkiej rurze według korelacji Blasiusa przy średniej prędkości płynu 0,015 m/s. Solver odpowiada, że korelacja obejmuje liczbę Reynoldsa od 4000 do 100 000. Wzorcowa odpowiedź mówi więc, że metoda nie obowiązuje, bo liczba Reynoldsa leży poniżej turbulentnego zakresu korelacji. Silnik ma w tej sprawie twardą umowę: nigdy NaN i nigdy cichy wynik poza zakresem ważności.

Trzecia sytuacja to pytanie, do którego nie ma narzędzia, bo dotyczy opisu, definicji albo historii. W naszych danych są to na przykład pytania o różnicę między dwoma rodzajami płuczki albo o historię wiertnictwa w Polsce. Model pisze wtedy `<f:brak_narzedzia>` i token odmowy. Odpowiedź w danych brzmi: „To pytanie nie wymaga obliczeń solverem; odpowiem na podstawie wiedzy ze źródeł.” Wiedzę ze źródeł model ma brać z dokumentów, o czym piszemy w następnym rozdziale. W pilotażu strojenia takie pytania stanowią 10% przykładów.

Co widać w odpowiedzi

Odpowiedź jest krótka i zawiera tylko liczby z bloku `<wynik>` albo z pytania. Pilnuje tego kontrola danych: przykład, w którego odpowiedzi stoi liczba spoza wyniku i pytania, odpada przed treningiem. Blok `<wynik>` ma najwyżej cztery wartości, podane w

jednostkach inżynierskich z czterema cyframi znaczącymi.

Modele 1B i 3B piszą prozą, bez tabel i list. To skutek tokenizera: jego normalizacja zamienia znak nowej linii na spację, więc w pretreningu modele nie widziały akapitów ani tabel. Tabele i listy ma dlatego składać program wykonawczy z bloku `<wynik>`. Solver oddaje też `kroki`, czyli nazwane wartości pośrednie. Z nich człowiek odtworzy tok obliczeń bez zaglądania do kodu.

Czas odpowiedzi zależy od sprzętu. Na karcie graficznej pełna pętla bez pamięci klucz-wartość trwała 1-3 s na pytanie. Pamięć klucz-wartość przechowuje obliczenia dla tokenów już przeczytanych, więc model nie przelicza całego tekstu przy każdym nowym słowie. Skraca to pętlę około 2,7-2,9 raza, a tekst odpowiedzi zostaje identyczny: 30 na 30 pytań w każdym z dwóch zestawów testowych. Na zwykłym procesorze, w lokalnym pokazie, generowanie trwało od 24 do 136 s na pytanie.

Co Stanisław umie dziś

Model 1B woła pięć narzędzi: ciśnienie hydrostatyczne, stratę ciśnienia przepływu płuczki w rurze, dopasowanie modelu płynięcia cieczy do pomiarów z wiskozymetru, metodę minimalnej krzywizny i geometrię trajektorii przewiertu sterowanego pod przeszkodą (HDD). Rejestr liczy już 147 narzędzi, bo do tych pięciu doszły 142 komendy nowego silnika. Wszystkie dotyczą na razie wierceń. Dziedziną startową są szerzej technologie pozyskiwania węglowodorów, a docelowo Stanisław ma być wszechstronnym modelem inżynierskim.

Nowych komend wersja SFT8 nie znała: w teście dymnym nie wybrała trafnie żadnej z 20, czego się spodziewaliśmy. Pilotaż strojenia nauczył model 25 z nich, najczęstszych w pracy inżyniera i na egzaminach. Na pytaniach, których nie widział, model po pilotażu daje poprawny wynik w 94,8% przypadków, a przy brakującej danej trafnie dopytuje w 86%. Najczęściej myli komendy podobne do siebie, na przykład objętość w rurze z objętością w przestrzeni pierścieniowej. To na razie wynik pilotażu na 1B, a nie model oddany do pracy.

Komendy, których model nie zna, można już dziś wywołać bez modelu. W lokalnym pokazie na serwerze projektu działa formularz zbudowany z rejestru: wpisuje się dane, a solver oddaje wynik z krokami i źródłem wzoru. To skromniejsza forma pracy, ale rachunek jest ten sam.

Na pytanie „kim jesteś” Stanisław ma odpowiadać, że jest pierwszym polskim modelem inżynierskim zintegrowanym z solverami, nazwanym na cześć prof. Stanisława Stryczka z Wydziału Wiertnictwa, Nafty i Gazu AGH. Zestaw uczący tożsamości wchodzi do najbliższego strojenia. Fakty o Profesorze model ma brać z dokumentu, a nie z pamięci wag.

Rola inżyniera

Pierwszym zadaniem Stanisława jest pomoc przy realnym projekcie otworu wiertniczego. Podział ról jest jednak taki sam w każdym projekcie. Solver liczy szybko i bez błędów arytmetycznych, ale model może wybrać złe narzędzie, a za projekt odpowiada człowiek. Narzędzia mają inżynierowi ułatwić sprawdzenie, a nie je zastąpić.

Arkusz projektu otworu działa według dwóch reguł. Zmiana danych wejściowych, na przykład geometrii albo płuczki, przelicza wszystkie wyniki zależne. Brak danej daje listę `brakuje`, nigdy domysł. Dla danych, które mają typowe wartości, arkusz podaje osobną listę `zalozenia` z uzasadnieniem i źródłem. Inżynier widzi więc, co policzono z jego danych, a co przyjęto.

Część konwencji wymaga ludzkiej decyzji. Świder to narzędzie, które zwierca skałę, i co pewien czas trzeba go wymienić, wyciągając cały przewód z otworu. Przy wyborze świdra według kosztu jednego metra otworu czas zapuszczania i wyciągania przewodu liczymy łącznie na wszystkie świdry. Taką konwencję autor projektu pamięta z zajęć, ale nie ma jej w źródłach, i tak to opisaliśmy w komendzie. Podobnie jest z obciążnikami, czyli ciężkimi rurami nad świdrem, które dają mu nacisk. Współczynnik zapasu przy doborze ich liczby jest jawnym parametrem bez wartości domyślnej. Dla danych z jednego kolokwium daje 52 obciążniki przy 1,0, 60 przy 1,15 i 65 przy 1,25, a rozstrzygnięcie zostawiamy człowiekowi.

Źródła też bywają omylne. Budując solver, znaleźliśmy w przykładach podręcznikowych kilkadziesiąt rozbieżności. Część to pomyłki rachunkowe w samych książkach, na przykład mnożenie przez współczynnik wyporu zamiast dzielenia, część to literówki w stałych i błędy rozpoznawania tekstu. Wzór wchodzi do solvera dopiero wtedy, gdy odtwarza przykład ze źródła co do cyfry, z tolerancją wynikającą z zaokrągleń tego źródła. Jeden zgodny przykład nie potwierdza jednak wszystkich warunków fizycznych. Granice stosowania metody inżynier sprawdza sam.

ROZDZIAŁ 6

Wiedza z dokumentów

Dlaczego fakty trzymamy poza wagami

Model 1B, który zbudowaliśmy od zera, dobrze radzi sobie z gramatyką polszczyzny, ale wiedzy ogólnej ma mało. Na polskich testach wiedzy wypada na poziomie zgadywania. Gdyby miał odpowiadać z pamięci wag na pytania o skład konkretnego materiału albo o wynik konkretnego badania laboratoryjnego, pisałby zdania gramatycznie poprawne i merytorycznie zmyślone. Dlatego fakty trzymamy w dokumentach, a model uczymy, jak z nich korzystać.

Służy do tego technika RAG (z ang. retrieval-augmented generation): przed odpowiedzią program wyszukuje fragmenty dokumentów pasujące do pytania i wkłada je do kontekstu modelu. Model ma odpowiedzieć na podstawie tych fragmentów, wskazać numer fragmentu, z którego wziął odpowiedź, a gdy odpowiedzi tam nie ma, powiedzieć to wprost. Technika nie zależy od dziedziny: w miejscu naszych artykułów mogą stać instrukcje, raporty z badań albo normy zakładowe. Ta sama zasada dotyczy patrona: biogram prof. Stanisława Stryczka trafia do bazy dokumentów, żeby model brał szczegóły z kontekstu, a nie z pamięci.

Indeks

Bazą wiedzy jest dziś 2340 artykułów z siedmiu polskich czasopism naukowych, razem 55 mln znaków tekstu polskiego. Najwięcej, 1380 artykułów, pochodzi z „Nafty-Gazu". Artykuły dzielimy na fragmenty po około 600 znaków z tytułem. Każdy fragment dostaje wektor osadzenia, czyli liczbowy odcisk znaczenia, a obok powstaje zwykły indeks pełnotekstowy. Wyszło 139 721 fragmentów, z czego 106 764 po polsku. Drugi, w większości angielski zbiór liczy 253 174 fragmenty i służy tylko do odczytu.

Indeks powstaje bez modelu generującego tekst, w tempie około 25 fragmentów na sekundę. Przy budowie danych wyszukiwanie łączy podobieństwo znaczeniowe z wektorów osadzenia i zgodność słów z indeksu pełnotekstowego. Potem osobny model porządkujący, tak zwany reranker, wybiera dwanaście najlepszych fragmentów.

Jak uczymy odpowiadać i odmawiać

Dane do nauki odpowiadania z dokumentów budujemy metodą RAFT (z ang. retrieval-augmented fine-tuning), czyli strojeniem na przykładach z wyszukanymi fragmentami. Każdy przykład ma kontekst z kilku fragmentów, pytanie i odpowiedź wzorcową. W

części przykładów kontekst zawiera fragment z odpowiedzią, który nazywamy wyrocznią, oraz trzy fragmenty-pułapki o podobnym temacie. W pozostałych wyroczni nie ma. Wzorcem jest wtedy odmowa albo, gdy odpowiedź da się złożyć z pozostałych fragmentów, odpowiedź oparta na nich.

Pytania i odpowiedzi wzorcowe pisały dwa otwarte modele-nauczyciele uruchomione lokalnie, w tym Bielik 11B w wersji do poleceń. Kod pilnuje trzech warunków. Odmowa ma zawsze jedną formę. Odpowiedź musi wskazać numer wyroczni w nawiasie kwadratowym. Każda liczba w odpowiedzi musi stać w kontekście albo w pytaniu. Przykład, który łamie którykolwiek warunek, odpada.

Oto dwa rekordy z zestawu testowego. W pierwszym kontekst zawiera cztery fragmenty, a wyrocznią jest fragment [2], opisujący wpływ utleniacza na płuczkę wiertniczą. Lepkość plastyczna i granica płynięcia to dwa parametry, którymi opisuje się płynięcie takiej cieczy:

```
<pytanie>Jakie zmiany w parametrach reologicznych płuczki nr 1 następują po
dodaniu podchlorynu wapnia w ilości 0,5% objętości w temperaturze 20°C?
</pytanie>
<odpowiedz>Po dodaniu podchlorynu wapnia w ilości 0,5% objętości w
temperaturze 20°C, następuje zdecydowany spadek parametrów reologicznych
płuczki nr 1. Lepkość plastyczna ulega obniżeniu do 15 mPas, a granica
płynięcia do 8 lb/100 ft2. Filtracja płuczki ulega zwiększeniu do 3,4 cm3, a
pH spada do ok. 8,0. [2]</odpowiedz>
```

W drugim kontekst zawiera cztery fragmenty o rurach okładzinowych, czyli stalowych rurach zabezpieczających ściany otworu, o cementach do ich uszczelniania i o konstrukcji studni, ale żaden nie mówi o pytanym ujęciu wody. Wzorcem jest odmowa:

```
<pytanie>Jakie materiały zostały wykorzystane do konstrukcji kolumn rur
okładzinowych i filtrów w studniach ujęcia "Grodzisko" w południowej części
Łodzi?</pytanie>
<odmowa><odpowiedz>Nie wiem – podane dokumenty nie zawierają odpowiedzi.
</odpowiedz>
```

Oba to odpowiedzi wzorcowe z danych, a nie wypowiedzi Stanisława. Pokazują, czego model ma się nauczyć: przepisać liczby z jednostkami dokładnie tak, jak stoją w źródle, i wskazać to źródło, a przy braku odpowiedzi nie dopowiadać niczego z ogólnej wiedzy o rurach.

Trudny przypadek: odpowiedź złożona z innych dokumentów

Najtrudniejsze okazały się pytania, w których wyroczni nie ma w kontekście, a odpowiedź da się złożyć z pozostałych fragmentów. Nazywamy je pytaniami z pułapkami. Znalezienie fragmentu z gotowym zdaniem tu nie wystarczy. Model musi ocenić,

czy kontekst wystarcza do odpowiedzi. Na teście, który ma 1079 pytań, takich pytań jest 378, a błędy modelu skupiają się właśnie w nich.

Dobrze widać to w liczbach. Najlepsza dziś wersja do pracy z dokumentami, SFT8, niesłusznie odmawia w 15,1% pytań z pułapkami, a tylko w 2,0% pytań, w których wyrocznia jest w kontekście. Im mniej w kontekście fragmentów z tego samego źródła co wyrocznia, tym częściej model odmawia. Przy braku takich fragmentów odmawia w 26,5% przypadków, przy czterech w 10,9%.

Część winy leży po stronie danych. Około 8% odpowiedzi wzorcowych w pytaniach z pułapkami samo stwierdza, że w dokumentach czegoś nie ma, choć przykład jest oznaczony jako pytanie z odpowiedzią. Model uczy się więc z etykiet, które nawzajem sobie przeczą. Kolejna wersja danych ma te przypadki przeetykietować i dołożyć przykłady, które uczą oceny, czy kontekst wystarcza.

Pomaga też dłuższy kontekst. Wersja 1B z kontekstem 8192 tokenów, strojona między innymi na przykładach z dodatkowymi fragmentami-pułapkami, w teście z długim kontekstem cytowała wyrocznię w 43% odpowiedzi, a SFT8 z kontekstem 2048 tokenów w 29%. Na krótkim kontekście wypadała gorzej.

Próg odmowy

Przez kilka rund strojenia próbowaliśmy poprawić odmowy, zmieniając proporcje danych i dokładając nowe przykłady odmów. Cztery wersje modelu 1B ułożyły się na jednej krzywej. Im częściej model trafnie odmawia w trudnych przypadkach, od 53,5% do 85,9%, tym częściej odmawia także tam, gdzie odpowiedź jest w dokumentach, od 4,8% do 18,6%. Każdy kolejny krok był droższy. Ostatni dał 10 dodatkowych trafnych odmów za 56 dodatkowych zbędnych.

Rozbicie odpowiedzi pytanie po pytaniu pokazało, że wersje różnią się progiem, a nie umiejętnością rozróżniania. Najbardziej ostrożna wersja odrzuca 107 pytań z odpowiedzią, których SFT8 nie odrzuca. W drugą stronę takie pytanie jest jedno.

Decyzja „odpowiadam czy odmawiam” zapada na pierwszym tokenie po pytaniu: model pisze albo token odmowy, albo token odpowiedzi. Prawdopodobieństwo tokenu odmowy można więc odczytać i porównać z progiem, zamiast uczyć kolejny model. Sam SFT8 z takim progiem rozróżnia przypadki tak samo dobrze jak pozostałe wersje: AUC wynosi 0,918 wobec 0,910-0,920, a przedziały ufności różnic obejmują zero. AUC to miara od 0 do 1, w której 0,5 oznacza zgadywanie, a 1 bezbłędne oddzielenie przypadków do odmowy od przypadków z odpowiedzią.

> **Zasada.** Kompromis między trafną a zbędną odmową ustawia się progiem przy pracy modelu, a nie kolejnym strojeniem.

Próg ma swoją cenę. Odpowiedzi wymuszone zbyt wysokim progiem bywają zmyślone, bo model mówi wtedy coś, czego nie wie. Dla asystenta inżyniera zalecamy więc próg, przy którym model raczej odmawia, niż zgaduje. Inżynier, który dostanie odmowę, poszuka w innym źródle. Inżynier, który dostanie pewnie brzmiącą liczbę wziętą znikąd, może ją wpisać do projektu. Wartości domyślnej progu jeszcze nie ustaliliśmy.

ROZDZIAŁ 7

Skąd wiemy, że działa

Zasady pomiaru

Każdą umiejętność mierzymy osobno: język, terminologię, wywołania narzędzi, odpowiadanie z dokumentów i odmowy, pracę z długim kontekstem. Mierzymy zawsze na zbiorach odłożonych, czyli takich, których model nie widział w treningu. Zbiór testowy dla nowej umiejętności powstaje, zanim ruszy strojenie, które ma jej uczyć. Żadna z tych zasad nie zależy od dziedziny; wiertnicze są tylko przykłady.

Odłożenie trzeba sprawdzać, a nie zakładać. Pierwsza kontrola danych dla modelu 3B pokazała, że 15% dokumentów testu PolEval ma w korpusie treningowym tak długie wspólne fragmenty, że model mógłby je znać na pamięć. Odkażanie usuwa z treningu dokumenty, które dzielą z tekstem testowym wiele ciągów po trzynaście tokenów, w pierwszym przejściu co najmniej dziesięć różnych. Po tej kontroli dołożyliśmy drugie przejście, z progiem dopasowanym do długości tekstu testowego. Bieg 3B ruszył dopiero przy 0% skażenia na domenie i 1,1% na PolEvalu, a wyniki podajemy na czystym podzbiorze.

Modele porównujemy na tym samym sprzęcie i tym samym zestawie. Tam, gdzie mamy zapis każdej odpowiedzi, zestawiamy modele pytanie po pytaniu testem McNemara, który sprawdza, czy różnica w trafieniach nie jest przypadkiem. Przedziały ufności liczymy metodą bootstrap, czyli wielokrotnym losowaniem pytań ze zwracaniem. Sprawdzamy treść wyniku, a nie kod wyjścia programu. Nauczył nas tego własny błąd: jedna z ewaluacji „przeszła", bo program zakończył się bez błędu, a model przez cały czas generował bezsensowny tekst ze złym tokenizerem. Dziś skrypt oceny odmawia startu, gdy słownik tokenizera nie pasuje do modelu.

Język i terminologia

Język mierzymy w bitach na bajt (bpb, z ang. bits per byte): ile bitów informacji model potrzebuje średnio na jeden bajt tekstu, którego nie widział. Niżej znaczy lepiej. Miara nie zależy od tokenizera, więc pozwala porównać modele o różnych słownikach.

Terminologię sprawdzamy na parach zdań, które różnią się jednym terminem. W jednym zdaniu stoi termin przyjęty w polskiej literaturze fachowej, na przykład „świder" albo „szybkość ścinania", w drugim potoczny zamiennik albo kalka z angielskiego. Taki test da

się zbudować dla każdej specjalności, która ma ustaloną terminologię. Wynik to odsetek par, w których model daje wyższe prawdopodobieństwo zdaniu poprawnemu. Zgadywanie daje 50%.

TEST (CZYSTY PODZBIÓR)	STANISŁAW 1B	BIELIK 11B BAZOWY	BIELIK 11B INSTRUCT
PolEval, bpb (niżej lepiej)	0,924 (kontekst 8k) · 0,929 (2k)	0,830	0,940
Domena, bpb (niżej lepiej)	0,686 (8k) · 0,723 (2k)	0,616	0,679
Terminologia, 86 par (wyżej lepiej)	0,907	nie mierzono	0,872

Czysty podzbiór PolEvalu to 1082 dokumenty bez żadnego wspólnego ciągu 13 tokenów z danymi Stanisława. Wśród artykułów z dziedziny zupełnie czystych nie było, więc bierzemy 98 takich, które dzielą z tymi danymi mniej niż 10% takich ciągów.

Bazowy Bielik jest wyraźnie lepszy w modelowaniu języka: o około 0,10 bpb na PolEvalu i 0,07-0,11 bpb na tekstach branżowych. Wcześniejsza wersja naszej strony porównywała Stanisława tylko z Bielikiem w wersji Instruct i pokazywała przewagę 1B na PolEvalu. Poprawiliśmy to po zmierzeniu wersji bazowej tą samą metodą. Strojenie do poleceń najpewniej pogarsza bpb, więc uczciwą poprzeczką dla 3B jest Bielik bazowy. Przewaga 1B w terminologii ma przedział około ±6-11 punktów procentowych, bo par jest tylko 86. Przygotowaliśmy szerszy zestaw 512 par, który czeka na przegląd wydziałowy.

Na polskich testach wiedzy 1B wypada na poziomie zgadywania. W teście Belebele z wyborem odpowiedzi A-D uzyskał 23,3% przy 25% dla losowego wyboru. Sygnał pojawia się dopiero wtedy, gdy oceniamy tekst odpowiedzi, a nie literę: 33,1%. Nasza uprząż oceny odtwarza opublikowane wyniki bazowego Bielika 11B z dokładnością do 1,2 punktu procentowego, więc to wynik modelu, a nie błąd pomiaru. Gramatykę 1B ma prawie jak Bielik: w teście poprawności gramatycznej 98,4% wobec 98,9%.

Narzędzia

Przy narzędziach liczymy dwie rzeczy: czy model wybrał właściwe narzędzie i czy solver, wywołany z argumentami modelu, dał ten sam wynik co dla argumentów wzorcowych. Na pięciu narzędziach model 1B wybiera trafnie zawsze, a poprawny wynik osiąga w 96-100% przypadków w czterech zestawach testowych.

Dla porównania zbudowaliśmy też linię bazową bez żadnego treningu: proste reguły, które szukają w pytaniu charakterystycznych słów i wzorców liczb. Wybierała narzędzie trafnie w 93,8% przypadków, poprawny wynik dawała w 81%, a trafnie odmawiała w

51,5%. Duży otwarty model ogólny z 27 mld wag, z opisami tych samych narzędzi, dał poprawny wynik w 99,6% pytań testu v1. Model 1B po bazowym strojeniu narzędziowym (SFT v3) jest w wywoływaniu solvera prawie na tym samym poziomie.

Najsłabiej wypada narzędzie z pięcioma parametrami, geometria trajektorii przewiertu sterowanego pod przeszkodą (HDD). W zestawie z parametrami spoza zakresu treningowego dało 25 trafień na 33 pytania. Z tego wzięła się zasada budowy silnika: małe komendy, a przy arkuszach z kilkunastoma danymi komenda pełna i komendy składowe.

Dokumenty i odmowy

Odpowiadanie z dokumentów sprawdzamy na teście z 1079 pytań: 142 trudne pytania, na które w kontekście nie ma odpowiedzi, i 937 pytań z odpowiedzią. Liczymy trafne odmowy, zbędne odmowy, cytat wyroczni i liczby z kontekstu. Dwie ostatnie miary liczymy tylko wśród odpowiedzi, których model udzielił.

MIARA (TEST 1079 PYTAŃ)	SFT9	SFT8	SFT10	SFT10A
Trafne odmowy (142)	0,535	0,655	0,789	0,859
Zbędne odmowy (937)	0,048	0,073	0,126	0,186
Cytat wyroczni	0,762	0,788	0,782	0,794
Liczby z kontekstu	0,868	0,886	0,867	0,862

Tabela pokazuje krzywą wymiany opisaną w poprzednim rozdziale. Żadna wersja nie jest lepsza od innych we wszystkim. Do pracy z dokumentami zostawiliśmy SFT8, bo przy 12,6% i 18,6% zbędnych odmów wersje SFT10 i SFT10a zbyt często milczą, gdy odpowiedź leży w dokumentach. Różnice potwierdza parowanie: dla zbędnych odmów SFT8 wobec SFT10 p jest rzędu 10 do potęgi minus 10.

Słabe miejsca

Model 1B nie ma wiedzy ogólnej i nie udajemy, że ją ma. Wiedza ma przychodzić z dokumentów. Testy wiedzy powtórzymy na modelu 3B, po którym spodziewamy się wyników ponad poziomem losowym. Trudne odmowy w odpowiadaniu z dokumentów utknęły na 1B w okolicy 65% i liczymy na większą pojemność modelu oraz czystsze dane.

Niektóre zestawy dają wynik zbyt optymistyczny i piszemy to przy liczbach. Test parafraz pytań układał ten sam model-nauczyciel, który parafrazował dane treningowe. Wszystkie dawne pytania testowe, na które model ma odpowiedzieć „nie mam narzędzia", okazały się obecne także w danych strojenia. Dawna miara odmów narzędziowych

mierzyła więc przeciek, a nie umiejętność. Dziś każdy rekord testowy ma pole, które mówi, czy jego pytanie występuje dosłownie w danych strojenia, i podajemy wynik bez takich rekordów.

Strojenie potrafi też zepsuć to, co już działało. Jedna z wczesnych wersji, strojona głównie na dokumentach i z małą ilością danych narzędziowych, zapomniała schematu argumentów dopasowania reologii. Wybierała dobre narzędzie, ale pisała argumenty jak dla straty ciśnienia w rurze. Od tamtej pory stroimy jednym przebiegiem na całości danych, a nie kilkoma kolejnymi. Pomiar potwierdził, że takie wspólne strojenie wypada lepiej.

Pilotaż strojenia, który uczy 25 nowych komend, sprawdziliśmy tak samo: wynik nowych narzędzi, dopytania i to, czy model nie zapomniał starych umiejętności.

MIARA PO PILOTAŻU (MODEL 1B)	PRZED (SFT8)	PO (SFT11)
Nowe komendy, poprawny wynik solvera (620 pytań)	0,000	0,948
Nowe komendy, wybór narzędzia (802 pytania)	0,100	0,960
Trafne dopytanie o brakującą daną (102 pytania)	0,000	0,863
Komendy łatwe do pomylenia, poprawny wynik (36 pytań)	0,000	0,250
Stare narzędzia, poprawny wynik (zestaw v2, 200 pytań)	0,982	0,994
Tożsamość, trafne odpowiedzi (14 pytań)	0 z 14	13 z 14
Dokumenty: trafne odmowy (142)	0,655	0,754
Dokumenty: zbędne odmowy (937)	0,073	0,129

Nowe narzędzia weszły, stare zostały. Słabo wypadły komendy łatwe do pomylenia i to jest pierwsza rzecz do poprawy w danych. W pracy z dokumentami pilotaż przesunął model w stronę odmawiania: trafnych odmów jest więcej, ale zbędnych też, mniej więcej tyle co w wersji SFT10. Cytowanie źródła i przepisywanie liczb z kontekstu zostały na tym samym poziomie. To kolejny powód, żeby kompromis ustawiać progiem przy pracy modelu. Wyniki modelu 3B podamy po zakończeniu jego pretreningu.

Ocena na Wydziale

Liczby z automatycznych testów nie mówią, czy odpowiedź przyda się inżynierowi. Dlatego Stanisław ma być oceniany przez naukowców Wydziału Wiertnictwa, Nafty i Gazu AGH. Protokół i bank pytań są gotowe, oceny jeszcze się nie odbyły.

Bank pilotażowy ma 35 pytań z dziewięciu dziedzin wiercenia otworów, od hydrauliki i kontroli otworu, czyli zapobiegania erupcjom, po cementowanie, koszty i przeliczenia jednostek. Pytania mają cztery poziomy: rachunkowe, pojęciowe, projektowe z brakującymi danymi i takie, na które właściwą odpowiedzią jest odmowa. Do każdego pytania jest lista elementów oczekiwanych i lista błędów krytycznych. W pytaniu o kartę likwidacji erupcji, gdy podano tylko gęstość płuczki i głębokość mierzoną wzdłuż otworu, błędem krytycznym jest wpisanie zmyślonych odczytów ciśnienia. Drugim błędem krytycznym jest użycie tej głębokości zamiast pionowej bez sprawdzenia, bo w otworze odchylonym od pionu obie się różnią, a ciśnienie słupa cieczy zależy od pionowej.

Ocena odbywa się w ciemno. Oceniający dostaje odpowiedzi Stanisława i modelu porównawczego bez nazw, na te same pytania, z tym samym kontekstem i tymi samymi narzędziami. Każdą odpowiedź ocenia w skali 0-3 osobno w czterech wymiarach: poprawność, przydatność dla inżyniera, terminologia i uczciwość co do braków. Zgodność między oceniającymi liczymy ważoną kappą Cohena, osobno dla każdej pary oceniających i każdego wymiaru. Wniosków o gotowości modelu do pracy przy projekcie nie wyciągniemy z samej małej próby rachunkowej.

ROZDZIAŁ 8

Odtworzenie krok po kroku w skrócie

Pełna instrukcja odtworzenia Stanisława od zera jest na stronie stanislaw.info, w zakładce „Instrukcja odtworzenia". Ma jedenaście etapów. Każdy opisuje cel, wejście, komendę ze skryptu w repozytorium projektu, wyjście, czas na naszym sprzęcie, sposób sprawdzenia i znane pułapki. Napisaliśmy ją dla człowieka, który pracuje sam, bez asystenta AI. Gdzie czasu nie zmierzyliśmy, piszemy to wprost.

Każdy skrypt kończy się znacznikiem w dzienniku, na przykład `TRENING_GOTOWY`. Sprawdzenie etapu polega na odczytaniu znacznika i porównaniu liczb w plikach statystyk z planem. Sam fakt, że program skończył pracę, niczego nie dowodzi.

Sprzęt i dane

Etap pierwszy to sprzęt i środowisko. Pracujemy na dwóch maszynach o różnych rolach. Serwer z ośmioma kartami RTX PRO 6000 Blackwell po 96 GB służy do pretreningu i strojenia w formacie bf16. Serwer wydziałowy z kartami Tesla V100 liczy ewaluacje, solvery, indeks dokumentów i parafrazy w formacie fp16. Oba formaty zapisują liczbę na 16 bitach i różnią się zakresem. Starsze karty V100 nie obsługują bf16, więc uczymy na nich w fp16.

Środowisko sprawdza test dymny: model z miliona wag przechodzi całą ścieżkę od pretreningu przez strojenie do oceny, a sam pretrening na 98 mln tokenów trwa około 4 minut. Wersji pakietów na serwerze treningowym nie zapisaliśmy.

Etap drugi to zebranie korpusów: polskiego internetu z publicznych zbiorów danych, dokumentów PDF z nauki i techniki, kuratorowanego korpusu polszczyzny, archiwum internetowego HPLT, artykułów z czasopism naukowych oraz tekstów branżowych. Przed deduplikacją było to 1175 plików, nie licząc tekstów branżowych. Teksty branżowe, 210 książek i 59 skanów, są chronione prawem autorskim. Używamy ich na podstawie przepisu o eksploracji tekstów do badań naukowych i ich nie publikujemy. Czasu pobierania nie zapisaliśmy, bo zależy od łącza. Surowe pobrania skasowaliśmy po deduplikacji, więc przy odtwarzaniu trzeba je pobrać ponownie.

Etap trzeci to tokenizer, czyli słownik kawałków słów, na które model tnie tekst. Nasz ma 32 768 pozycji i powstał na próbce około 2 mld znaków z dziesięciu źródeł, z tekstami branżowymi wziętymi sześciokrotnie. Chodziło o to, żeby terminy fachowe, takie jak „ciśnienie" czy „świder", mieściły się w jednym tokenie. Tokenizer ma 69 symboli specjal-

nych: 13 do struktury rozmowy, 6 do narzędzi i 50 zapasowych. Na tekstach odłożonych tnie polski internet dobrej jakości na kawałki średnio po 4,78 znaku, wobec 4,06 w poprzedniej wersji.

Etap czwarty to deduplikacja między źródłami. Na 64 rdzeniach trwała 2,3 h. Ten sam tekst trafia do kilku zbiorów, a chcemy, żeby każdy dokument liczył się raz i żeby żaden tekst testowy nie wrócił do treningu jako kopia z innego źródła. Szukamy więc kolejno identycznych tekstów, tych samych adresów stron i tekstów prawie identycznych. Tych ostatnich szukamy przez odciski zbiorów pięciowyrazowych zbitek, metodą MinHash. Z każdej grupy duplikatów zostaje kopia z najcenniejszego źródła.

Liczba dokumentów spadła z 213,6 do 179,0 mln, a liczba znaków z 925 do 715 mld, czyli do ok. 161 mld tokenów obecnego tokenizera. Na dysku korpus po deduplikacji zajmuje 243 GB. W losowej kontroli 20 usuniętych par 18 było prawdziwymi duplikatami, a dwie miały ten sam adres i inną treść. Reguła adresu okazała się za ostra i zabrała ok. 1,62 mln dokumentów, które trzeba przywrócić.

Etap piąty buduje z korpusu pliki treningowe w zadanych proporcjach źródeł i odkaża je względem testów. Dla modelu 3B odkażona część treningowa ma 55,3 mld tokenów, a część do fazy wyżarzania 5,3 mld; przed odkażaniem było to 60,4 i 6,2 mld. Odkażanie usuwa z treningu dokumenty, które dzielą z testami długie ciągi tokenów. Bramka przed startem 3B wymagała 0% skażonych dokumentów domeny i poniżej 1% w teście PolEval. PolEval wyszedł 1,1%, o 0,1 punktu ponad progiem; uznaliśmy dane za czyste, a wyniki na PolEvalu podajemy na czystym podzbiorze. Mała mieszanka na 2 mld tokenów budowała się około 6 minut. Czasu trenowania tokenizera i budowy pełnych mieszanek nie zapisaliśmy.

Pretrening

Etap szósty to pretrening, czyli nauka przewidywania następnego tokenu na surowym tekście. Trener mieści się w jednym pliku. Harmonogram szybkości uczenia ma trzy fazy: rozgrzewkę, długi odcinek stałej szybkości i wyżarzanie w ostatnich 10% kroków, kiedy szybkość spada liniowo, a model dostaje staranniej wybrane dane. Model 1B ma 1,198 mld wag. Uczył się 41 h na ośmiu kartach na około 28 mld tokenów. Model 3B ma 3,045 mld wag i uczy się na 60 mld tokenów, w tempie około 77,6 tys. tokenów na sekundę, co daje około dziewięciu dni na ośmiu kartach.

Bieg 3B zapisuje punkt kontrolny co 200 kroków, a każdy taki zapis ze stanem optymalizatora ma 36,5 GB. Biegu pilnuje czujka, czyli program, który co minutę sprawdza procesy, stratę i walidację. Gdy procesy padną, czujka sama wznawia bieg z ostatniego punktu kontrolnego, najwyżej trzy razy, a sprawdzony punkt trzyma na wypadek ręcznego powrotu. Gdy strata wygląda źle, nie wznawia niczego i zostawia decyzję człowiekowi.

Etap siódmy rozszerza kontekst modelu 1B z 2048 do 8192 tokenów. Model uczy się dalej przez około 7 h na dwóch kartach, na około 1 mld tokenów z przewagą długich dokumentów. Sprawdzianem jest test „hasła w stogu siana": pięciocyfrowe hasło ukryte w długim dokumencie, które model ma przytoczyć. Przed rozszerzeniem model znajdował hasło zawsze przy 2048 tokenach i nigdy przy 4096 i 8192. Po rozszerzeniu znajduje je w 97,5% prób przy 4096 i 8192 tokenach, a przy 2048 nadal zawsze.

Strojenie i ocena

Etap ósmy przygotowuje dane do strojenia. Są trzy rodzaje: wywołania narzędzi z wynikiem policzonym przez solver, parafrazy pytań i przykłady odpowiadania z dokumentów. Generatory solverowe to czysty kod i liczą minuty. Parafrazy i przykłady z dokumentami piszą lokalne modele-nauczyciele z otwartymi wagami, uruchomione na serwerach, z których korzystamy. Walidatory w kodzie odrzucają złe wyjścia: parafraza odpada, gdy zmieniła liczby, zgubiła jednostkę albo urosła ponad trzykrotnie. Odsetek przyjętych parafraz około 67% to norma.

Etap dziewiąty to strojenie. Strata liczy się od pierwszego tokenu po pytaniu, czyli od wywołania narzędzia albo od odpowiedzi. Blok z wynikiem solvera jest z niej wyłączony, bo w działaniu wstawia go solver, a nie model. Bazowe strojenie narzędziowe 1B (SFT v3) trwało około 35 minut na dwóch kartach, a strojenie SFT10 na dwóch kartach V100 4 h 37 min. Najważniejsza lekcja tego etapu brzmi: jedno wspólne strojenie na całości danych daje lepszy model niż kilka strojeń po kolei. Czasu generowania parafraz i przykładów z dokumentami ani strojeń SFT6 i SFT8 nie zapisaliśmy.

Etap dziesiąty to ewaluacja, opisana w rozdziale 7. Ocena narzędzi idzie dwuetapowo. Model generuje wywołania tam, gdzie leżą jego wagi, a solver ocenia je tam, gdzie działa, bo przeniesienie jednego pliku wag wolnym łączem między serwerami trwało około godziny. Linia bazowa Bielika na polskim teście wiedzy liczyła się 68 minut, pozostałe oceny trwają od minut do godzin.

Lekcje

Etap jedenasty to lista min, na które weszliśmy. Kilka z nich warto znać przed startem w każdej dziedzinie.

Program, który rozdzielał pracę na 80 procesów z całym korpusem w pamięci, skopiował go w każdym procesie i na około 10 minut unieruchomił serwer. Od tamtej pory ciężkie zadania mają twardy limit pamięci. Skrypt bez kontroli błędów wpisał znacznik sukcesu po padzie treningu, więc każdy krok krytyczny ma dziś własny znacznik błędu.

Nowa architektura modelu, uruchomiona przez ogólną bibliotekę w fp16 na starszych kartach, zablokowała wszystkie karty serwera aż do restartu. Zanik zasilania przerwał jedno strojenie przy kroku 950 z 4000. Wznowiliśmy je z punktu zapisanego po kroku 500 i straciliśmy tylko pracę między zapisami.

ROZDZIAŁ 9

Plan: 3B, 7B i dalej

Kierunek

Docelowo Stanisław ma być wszechstronnym modelem inżynierskim. Zaczynamy od wiedzy, na której znamy się najlepiej, czyli od szeroko pojętych technologii pozyskiwania węglowodorów. Tam mamy podręczniki, solvery i ludzi, którzy potrafią sprawdzić wynik. To, co już zbudowaliśmy i zmierzyliśmy, dotyczy na razie wierceń. Metoda nie zależy od dziedziny: dokumenty zamiast pamięci wag, solver z bramką, rejestr narzędzi i uczciwy pomiar. Planów ani terminów dla kolejnych dziedzin w tym wydaniu nie podajemy.

Dokończyć i ocenić 3B

Pretrening modelu 3B ma się skończyć około 19 listopada 2026 roku. Walidacja na odłożonej części danych spadła z 1,257 bpb po 1000 krokach do 0,844 bpb po 6000 krokach. Tej krzywej nie porównujemy wprost z krzywą 1B, bo oba modele mają inne zbiory walidacyjne. Gotowy model ocenimy na czystych testach, a poprzeczką będzie bazowy Bielik 11B, który w modelowaniu polszczyzny jest dziś wyraźnie lepszy od 1B. Do oceny dochodzą polskie testy Belebele i INCLUDE.

Fazę wyżarzania 3B przeprowadzimy w dwóch wariantach. Wariant A idzie na danych polskich, tak jak zaplanowano. Wariant B rusza z tego samego punktu kontrolnego sprzed wyżarzania, po kroku 51 200, na mieszance 70% polskiego i 30% angielskiego. To 3,86 mld tokenów polskich i 1,65 mld angielskich, w tym angielskie podręczniki branżowe, odkażone względem testów. Oba warianty porównamy na tych samych testach. Obecny tokenizer jest polski. Angielskie teksty naukowe z plików PDF kosztują w nim 2,13 tokenu na słowo, a polskie teksty z sieci 1,57, więc angielski wymaga o ok. jedną trzecią więcej obliczeń.

Strojenie 3B

Po pretreningu 3B dostanie strojenie z całym rejestrem narzędzi, czyli ze wszystkimi komendami, które przeszły bramkę. Sprawdziliśmy już, ile to kosztuje. Próbne strojenie na wagach pośrednich, z kroku 9200, na czterech kartach Tesla V100 było stabilne w fp16. Strata spadła z 5,16 do 0,17 w 200 krokach przy 4,23 s na krok, więc pełne 4000 kroków zajmie około 4,9 h. Najpierw jednak przepis pilotażowy z modelu 1B przechodzi na pośrednie wagi 3B. Warunkiem było to, żeby na 1B nauczył nowych narzędzi i nie

zepsuł starych. Ten warunek pilotaż spełnił, a w pracy z dokumentami przesunął model w stronę odmawiania, co opisujemy w rozdziale 7. Strojenie pośrednich wag 3B ruszyło 12 listopada.

Przed pełnym strojeniem poprawiamy dane. Liczby w pytaniach mają wyglądać jak w praktyce: średnice z typoszeregów rur i świdrów, wartości zaokrąglone tak, jak zaokrągla je inżynier, część zapisana ułamkami. Dopytania mają podawać jednostki, w których człowiek zna daną wielkość. Stare i nowe narzędzia trzeba ujednolicić tak, żeby jeden rachunek miał jedną nazwę, bo model, który widzi dwie etykiety na to samo pytanie, zaczyna je mylić. Nazw komend i pól po starcie treningu nie zmieniamy. Każda zmiana takiej umowy rozjechałaby dane treningowe z rejestrem.

Równolegle przygotowujemy dane dwujęzyczne: 23 518 rekordów, w tym 11 759 par odpowiedzi polskich i angielskich na te same zadania. Stanisław ma rozumieć pytanie po angielsku, odpowiadać domyślnie po polsku, a po angielsku tylko na wyraźną prośbę.

Czy i kiedy LoRA

LoRA to sposób strojenia, w którym wagi modelu zostają zamrożone, a uczy się mały dodatek do nich, zwany adapterem. Adapter ma ułamek rozmiaru modelu. Strojenie potrzebuje więc mniej pamięci, a jego wynikiem jest mały plik, który można do modelu dołożyć albo z niego zdjąć.

Dziś stroimy pełne wagi i mamy ku temu trzy powody. Pełne strojenie mieści się na naszym sprzęcie, co pokazała opisana wyżej próba na modelu 3B. Pierwsze strojenie jest dla modelu trenowanego od zera dużą zmianą, bo dopiero w nim poznaje on format rozmowy i tokeny narzędzi, a adapter z założenia zmienia model niewiele. Odruch sięgania po solver ma wreszcie siedzieć w samych wagach, a nie w dodatku.

Rachunek zmieni się przy modelu 7B. Pełne strojenie trzyma w pamięci cztery liczby na każdy parametr: wagę, gradient i dwa stany optymalizatora. W pojedynczej precyzji to 16 bajtów na parametr, czyli ponad 110 GB dla siedmiu miliardów wag, zanim policzymy cokolwiek poza nimi. Na kartach po 32 GB, na których robimy próby strojenia, zrobi się ciasno, a serwer z większymi kartami będzie wtedy tygodniami zajęty pretreningiem. Z zamrożoną bazą i adapterem pośrednie wersje 7B da się stroić na mniejszej liczbie kart.

Drugie miejsce dla adapterów to pętla doskonalenia opisana niżej. Wersja oceniona przez naukowców zostaje nietknięta, a kolejne poprawki przychodzą jako małe adaptery, każdy z testem przed i po. Cofnięcie nieudanej poprawki oznacza zdjęcie adaptera. Co pewien czas zebrane poprawki wtapiamy w wagi pełnym strojeniem. Adapter przyda się też przy uczeniu z nagrodą liczoną przez solver, bo takie uczenie bywa niestabilne i dobrze jest móc tanio wycofać nieudaną próbę.

To jest plan, a nie wynik. Zanim przejdziemy na adaptery, zmierzymy różnicę: ten sam pilotaż strojenia w obu wariantach, na tych samych danych i testach, z pomiarem tego, ile model przy okazji zapomniał.

7B

O starcie modelu 7B zdecydujemy po ocenie 3B. Przygotowania są gotowe, ale nic nie było trenowane w pełnej skali.

Pierwsza decyzja dotyczy tokenizera, bo jego zmiana oznacza pretrening od nowa. Proponujemy nowy słownik z 65 536 pozycjami, uczony na próbce z 20% tekstu angielskiego i zachowujący znaki nowej linii. Modele 1B i 3B nigdy nie widziały akapitów, list ani tabel, bo obecny tokenizer zamienia nową linię na spację. Nowy tokenizer tnie polski o 1,2%, a angielski o 30% oszczędniej niż obecny. Za zachowanie nowych linii płacimy 4-9% tokenów.

Tokeny sterujące, takie jak początek kontekstu czy wywołania, mają w nim dostać typ, którego zwykły tekst nie może wytworzyć. Dziś dokument z bazy wiedzy, który zawiera napis wyglądający jak taki znacznik, zamienia się w prawdziwy token sterujący i model może go wziąć na przykład za początek wywołania narzędzia.

Model 7B nie mieści się w obecnym schemacie uczenia, w którym każda karta trzyma pełną kopię wag i stanu optymalizatora. Same stany zajęłyby 121,5 GiB na kartę, która ma 95,6 GiB. Nowy trener dzieli wagi, gradienty i stany optymalizatora między karty, a zamiast przechowywać wyniki pośrednie każdej warstwy, liczy je ponownie, gdy są potrzebne. Według naszego modelu pamięci, skalibrowanego na biegu 3B, na kartę przypadnie wtedy ok. 64 GiB. Trener sprawdziliśmy na małym modelu: w trybie deterministycznym bieg przerwany i wznowiony z punktu kontrolnego daje co do bitu ten sam wynik co bieg ciągły.

Budżet czasu liczymy z mocy zmierzonej na biegu 3B. Model 7B na 100 mld tokenów to około 49 dni na obecnym serwerze treningowym. Na 60 mld tokenów skróciłby się do około 29 dni, ale dostałby tylko około 8 tokenów na wagę, daleko od około 20 z reguły Chinchilli, i byłby niedouczony. Alternatywą jest model około 5B na 85-100 mld tokenów w 4-5 tygodni. Przed decyzją przeprowadzimy krótką sondę na serwerze treningowym, przygotowaną na około 40-55 minut. Zmierzy ona przepustowość połączeń między kartami, zysk z obliczeń w FP8, czyli na liczbach ośmiobitowych, i rzeczywistą pamięć 7B. Dziś te wartości są założeniami.

Polskiego tekstu wystarczy. Po deduplikacji polski korpus ma ok. 715 mld znaków. W obecnym tokenizerze to ok. 161 mld tokenów, a w proponowanym tokenizerze z zachowaniem nowych linii ok. 159 mld. Budżet 100 mld tokenów w proporcji 75 do 25 dla polskiego i angielskiego to mniej więcej połowa jednej epoki polszczyzny.

Wąskim gardłem jest domena. Polskich podręczników branżowych mamy około 13 mln tokenów, a tabele są w nich zapisane surowym HTML-em. Przed 7B trzeba je przepisać do zwięzłej postaci, przyciąć pętle po błędach rozpoznawania tekstu i rozpoznawać język po słowach, a nie po polskich znakach.

Licencje części norm technicznych wykluczają użycie ich tekstu w modelach. Z norm bierzemy więc tylko wzory i liczby do solvera, a tekst zostaje poza bazą wiedzy i treningiem.

Odruch solvera i pętla doskonalenia

Celem wszystkich etapów jest odruch: model ma sięgać po solver, zanim policzy cokolwiek sam. Na 1B sprawdziliśmy przepis pilotażowy na 25 komendach. Na 3B stroimy z całym rejestrem. Przy 7B chcemy wpisać odruch w wagi bazowe: ślady wywołań narzędzi, od pytania przez wywołanie i wynik po odpowiedź, mają stanowić kilka procent tokenów fazy wyżarzania pretreningu. Potem planujemy uczenie z nagrodą liczoną przez solver. Poprawność wyniku da się wtedy sprawdzić bez sędziego, bo solver zna właściwą odpowiedź.

Rejestr rośnie także między rundami strojenia. Częste komendy model ma znać na pamięć, a nowe ma obsłużyć z ich opisu wstawionego do kontekstu i dopiero w kolejnej rundzie strojenia przyjąć je do wag. Opisy wszystkich 147 narzędzi w zwięzłym zapisie to około 20 tys. tokenów. W kontekście 4096 tokenów się nie zmieszczą, więc przed pytaniem trzeba wybrać kilka do kilkunastu kandydatów. Przygotowaliśmy zestaw ćwiczebny z 320 nazwami narzędzi i 40 operacjami, w którym część rodzin narzędzi model zobaczy tylko w teście. Wyniku modelu na tym zestawie jeszcze nie mamy. Odruch będziemy mierzyć dwiema liczbami: odsetkiem pytań obliczeniowych, w których model liczy sam zamiast wołać solver, z celem równym zeru, oraz liczbami w odpowiedzi, których nie ma w wyniku ani w pytaniu.

Każda runda doskonalenia ma ten sam rytm. Zbieramy błędy, zapisując każdą odpowiedź testową z cechami pytania. Oceniamy je i poprawiamy etykiety. Budujemy dane, które przechodzą bramkę kontroli przed treningiem. Robimy krótkie strojenie najpierw na małym modelu. Porównujemy wersję nową ze starą na tych samych testach.

Nowa wersja ma opanować nowe umiejętności i nie zepsuć starych, a dopóki tego nie pokaże, w użyciu zostaje poprzednia. Do pracy z dokumentami wciąż służy SFT8, bo późniejsze wersje przegrały na zbędnych odmowach.

Dalej

Plan zależy od sprzętu. Gdyby pojawił się serwer z większymi kartami, o którym rozmawiamy, bieg taki jak obecny 3B trwałby tam 1,5-2 dni, a model z 8 mld wag na 250 mld tokenów około 17 dni. To szacunki przy założonej mocy, której nie zmierzyliśmy.

Kilka rzeczy przygotowujemy już teraz, bo później byłyby droższe. Zbieramy pary rysunek i podpis z podręczników, takie jak nomogramy, wykresy i schematy otworów, wyłącznie do własnej oceny i nauki w granicach prawa. Decyzje modelu, czyli odmowa, dopytanie i wybór narzędzia, chcemy wyprowadzić jako skalibrowane prawdopodobieństwa, tak jak dziś próg odmowy. Do szybszego generowania planujemy mały model pomocniczy z tym samym tokenizerem, który podpowiada kolejne tokeny dużemu.

ROZDZIAŁ 10

Własny model w twojej organizacji

Zacznij od tego, na czym znasz się najlepiej

Pierwsza decyzja dotyczy dziedziny. Model wybiera się później. Zacznij od wiedzy, na której twój zespół zna się najlepiej. Tam masz dokumenty, obliczenia, które da się sprawdzić, i ludzi, którzy odróżnią dobry wynik od złego. Bez tych trzech rzeczy nie zbudujesz ani solvera, ani testu, ani uczciwej oceny.

My zaczęliśmy od technologii pozyskiwania węglowodorów, a w nich od wierceń, bo tam mamy podręczniki, zadania z kolokwiów i naukowców Wydziału, którzy ocenią odpowiedzi. Metoda nie jest jednak wiertnicza. Dokumenty zamiast pamięci wag, solver z bramką, rejestr narzędzi i pomiar na odłożonych testach nie zależą od dziedziny i powinny działać tak samo dla konstrukcji stalowych, sieci ciepłowniczych czy maszyn wirnikowych. Sprawdziliśmy je na razie w wierceniach. Gdy metoda zadziała w jednej dziedzinie, przenosisz ją na następną.

Rozłóż problem na trzy części

Pytania inżyniera do modelu językowego dzielą się na trzy rodzaje. Pierwszy to rachunek: ciśnienie, naprężenie, objętość, koszt. Tego nie powinien robić model, tylko deterministyczny program, który dla tych samych danych zawsze da ten sam wynik. Drugi to wiedza: co mówi norma zakładowa, jaki był wynik badania, jak opisano procedurę. To należy do dokumentów, z których model cytuje źródło. Trzeci to zachowanie: czy model sięgnie po narzędzie, czy dopyta o brakującą daną, czy powie „nie wiem". Tylko to ostatnie trzeba wpisać w wagi.

Ten podział ma praktyczny skutek. Solver i indeks dokumentów są warte swojej pracy niezależnie od tego, czyj model z nich korzysta. Solver można poprawić albo wymienić bez treningu. Indeks uzupełnia się nowym dokumentem, również bez treningu. Jeśli zaczniesz od nich, zmiana modelu nie przekreśli tej pracy.

Solver buduj z bramką. Wzór wchodzi do programu dopiero wtedy, gdy odtwarza przykład z podręcznika albo z własnej dokumentacji co do cyfry. Błąd danych ma dawać komunikat po ludzku, nigdy cichy wynik spoza zakresu ważności metody. Sprawdzaj przy tym samo źródło. U nas bramka wykryła w przykładach podręcznikowych kilkadziesiąt rozbieżności, w tym zwykłe pomyłki rachunkowe w książkach.

Kiedy wystarczy gotowy model

Często wystarczy gotowy model z otwartymi wagami, uruchomiony na własnym serwerze, z twoimi dokumentami i twoim solverem. Dane nie opuszczają wtedy organizacji, a ty nie płacisz za pretrening. W naszym teście duży otwarty model ogólny z 27 mld wag, któremu podaliśmy opisy pięciu narzędzi, dał poprawny wynik solvera w 99,6% przypadków. Jeśli twoje potrzeby to głównie rachunek i dokumenty, zacznij od takiego układu i zmierz go.

Zanim cokolwiek wytrenujesz, zbuduj linię bazową bez treningu. Nasza linia bazowa, czyli proste reguły szukające w pytaniu słów i wzorców liczb, trafnie wybierała narzędzie w 93,8% przypadków. Dopiero na tle takiej liczby widać, ile daje model. Czasem daje mniej, niż się wydaje. Czasem okazuje się, że proste reguły albo gotowy model wystarczą i strojenie mija się z celem.

Kiedy własny model ma sens

Własny model ma sens, gdy zachowanie i język są ważniejsze niż wiedza ogólna. Mały model, strojony na waszych danych, może dorównać dużemu w wąskim zadaniu. U nas model 1B po bazowym strojeniu narzędziowym (SFT v3) wołał solver prawie tak dobrze jak model z 27 mld wag. W teście terminologii fachowej wypadł lepiej od większego modelu ogólnego, choć ta przewaga mieści się w niepewności pomiaru. Mały model działa też na skromnym sprzęcie. Nasz 1B odpowiadał na karcie graficznej w 1-3 s, a na zwykłym procesorze w 24-136 s.

Model od zera daje pełną kontrolę. Twoje są dane, tokenizer z tokenami specjalnymi dla waszych narzędzi i to, jak model się przedstawia. Wiesz dokładnie, co model widział, więc wiesz, które testy są czyste. Dla uczelni i działu badawczego to także sposób na zrozumienie, jak taki model działa od środka.

Trzeba jednak znać koszt. Nasz model 1B uczył się 41 h na ośmiu kartach, 3B uczy się około dziewięciu dni, a 7B na 100 mld tokenów zająłby około siedmiu tygodni. Do tego dochodzi czas ludzi: zebranie i deduplikacja korpusu, odkażanie testów, dane do strojenia i ich kontrola. Mały model od zera nie będzie miał wiedzy ogólnej. Nasz 1B wypada na testach wiedzy na poziomie zgadywania i właśnie dlatego wiedzę trzymamy w dokumentach.

Sprawdź też prawa do danych. Teksty chronione prawem autorskim wolno nam eksplorować tylko w celach badawczych, a licencje części norm technicznych wykluczają użycie ich tekstu w modelach. Z takich norm wolno nam wziąć wzory i liczby do solvera, ale nie tekst do bazy wiedzy. W twojej organizacji zasady mogą być inne, więc sprawdź je, zanim zaczniesz zbierać korpus.

Jeśli masz wybór między modelem od zera a dalszym uczeniem gotowej otwartej bazy, nie rozstrzygaj tego z góry. Zmierz oba warianty na tej samej ocenie i wybierz po liczbach.

Od czego zacząć

Proponujemy taką kolejność:

1. Wybierz dziedzinę, na której znasz się najlepiej, i ludzi, którzy ocenią odpowiedzi.
2. Spisz obliczenia wykonywane najczęściej i do każdego znajdź przykład z kompletem danych i wynikiem.
3. Zbuduj solver z bramką: każdy wzór odtwarza przykład co do cyfry, a błąd danych daje komunikat zamiast liczby.
4. Zrób rejestr narzędzi: nazwa, jednozdaniowy opis, pola z jednostkami i zakresami.
5. Zbierz dokumenty, sprawdź prawa do nich i zbuduj indeks, z którego model cytuje źródło.
6. Odłóż zestawy testowe przed jakimkolwiek treningiem i sprawdź, czy nie ma ich w danych.
7. Zmierz linię bazową bez treningu i gotowy model z narzędziami.
8. Dopiero wtedy zdecyduj o strojeniu albo o własnym modelu.

Czego się wystrzegać

Sprawdzaj treść wyniku, a nie to, że program skończył się bez błędu. Nasza ewaluacja „przeszła", choć model z niewłaściwym tokenizerem przez cały czas generował bezsensowny tekst.

Nie układaj testu tym samym narzędziem, którym układasz dane treningowe. Gdy ten sam model pisał parafrazy do treningu i do testu, wynik testu był zbyt optymistyczny. Porównuj z właściwym punktem odniesienia. My najpierw zestawiliśmy swój model z innym modelem tylko w wersji strojonej do poleceń i wyszła nam przewaga. Wersja bazowa, zmierzona tą samą metodą, okazała się od naszego modelu wyraźnie lepsza, więc to ona jest uczciwą poprzeczką. Poprawiliśmy to publicznie.

Nie stroj modelu od nowa za każdym razem, gdy chcesz przesunąć kompromis. Nasze cztery wersje różniły się progiem odmowy, a nie umiejętnością rozróżniania, i ten sam efekt dał próg ustawiony przy pracy jednego modelu. Konwencje, których nie ma w źródłach, opisuj w narzędziu jawnie, a decyzje, których źródła nie rozstrzygają, zostaw człowiekowi.

Słowniczek

Hasła z uczenia maszynowego i z naszej metody oraz kilka terminów wiertniczych, które w książce wracają najczęściej, objaśnionych dla osoby spoza branży. Kolejność alfabetyczna.

- *AUC*: miara, która mówi, jak dobrze jedna liczba, na przykład prawdopodobieństwo odmowy, oddziela przypadki dwóch rodzajów. Wartość 0,5 oznacza zgadywanie, 1 rozdzielenie bez błędu.
- *bpb (bity na bajt)*: średnia liczba bitów, której model potrzebuje na zapisanie jednego bajtu tekstu, którego nie widział. Niżej znaczy lepiej, a miara nie zależy od tokenizera, więc pozwala porównać różne modele.
- *bramka*: warunek wejścia wzoru do solvera. Test musi odtworzyć przykład ze źródła co do cyfry, a wzór bez zaliczonego testu nie istnieje.
- *czujka*: program pilnujący wielodniowego treningu. Co minutę sprawdza procesy, stratę i walidację, a po awarii sama wznawia bieg z ostatniego punktu kontrolnego.
- *deduplikacja*: usuwanie z korpusu dokumentów powtórzonych i prawie powtórzonych, żeby każdy dokument liczył się raz, a kopie tekstów testowych nie wróciły do treningu z innego źródła.
- *dekodowanie z ograniczeniem*: generowanie tekstu, w którym program dopuszcza tylko tokeny zgodne z regułą. U nas wywołanie może zawierać wyłącznie nazwę z rejestru, pola z opisu narzędzia i liczby obecne w pytaniu.
- *dopytanie*: zamiast liczyć z niepełnych danych, model pyta o brakującą wielkość i podaje, w jakich jednostkach ją przyjmie.
- *epoka*: jedno pełne przejście przez wszystkie dane treningowe.
- *FSDP*: sposób uczenia na wielu kartach, w którym wagi, gradienty i stany optymalizatora są podzielone między karty. Potrzebny wtedy, gdy na jednej karcie nie mieszczą się wagi razem z gradientami i stanami optymalizatora.
- *hasło w stogu siana*: test długiego kontekstu. Model ma przytoczyć krótkie hasło ukryte w różnych miejscach długiego dokumentu.
- *karta likwidacji erupcji*: arkusz obliczeń, którym opanowuje się niekontrolowany dopływ płynu ze skał do otworu wiertniczego. Zawiera między innymi gęstość płuczki potrzebną do zrównoważenia ciśnienia złożowego oraz ciśnienia cyrkulacji na początku i na końcu zabiegu.

- *kontekst*: tekst, który model widzi naraz, mierzony w tokenach. Modele 1B i 3B mają kontekst 2048 tokenów, a rozszerzona wersja 1B 8192.
- *korpus*: zbiór tekstów, na którym odbywa się pretrening.
- *krok*: jedna aktualizacja wag na partii danych. W naszym pretreningu jedna partia ma około 1,05 mln tokenów.
- *krzywa wymiany*: zależność, w której poprawa jednej miary kosztuje pogorszenie drugiej. U nas więcej trafnych odmów oznaczało więcej odmów zbędnych.
- *LoRA*: sposób strojenia, w którym wagi modelu zostają zamrożone, a uczy się mały dodatek do nich, zwany adapterem. Adapter można do modelu dołożyć albo z niego zdjąć.
- *maska straty*: wskazanie, które tokeny przykładu liczą się do straty. U nas strata liczy się od pierwszego tokenu po pytaniu: obejmuje wywołanie i odpowiedź, a pomija treść pytania i wynik solvera, który w działaniu wstawia program.
- *model-nauczyciel*: inny model językowy z otwartymi wagami, uruchomiony lokalnie, który pisze parafrazy i przykłady odpowiedzi z dokumentów. Jego wyjścia sprawdzają walidatory w kodzie.
- *odkażanie*: usuwanie z danych treningowych tekstów, które pokrywają się z testami. Czysty podzbiór testu to ta jego część, która dzieli z treningiem nie więcej niż ustaloną część ciągów tokenów: w PolEvalu żadnego, w artykułach z dziedziny mniej niż 10%.
- *odmowa*: odpowiedź „nie wiem” albo „nie mam do tego narzędzia”. Odmowa trafna pada wtedy, gdy odpowiedzi naprawdę nie ma, zbędna wtedy, gdy odpowiedź była w dokumentach.
- *pamięć klucz-wartość*: przechowane obliczenia dla tokenów już przeczytanych. Przyspiesza generowanie i nie zmienia jego wyniku.
- *płuczka wiertnicza*: ciecz krążąca w otworze wiertniczym. Wynosi na powierzchnię okruchy skały i swoim ciśnieniem równoważy ciśnienie płynów w skałach.
- *pretrening*: pierwsza i najdłuższa faza uczenia, w której model przewiduje następny token w surowym tekście.
- *przeciek*: sytuacja, w której pytanie testowe występuje dosłownie w danych strojenia. Zawyża wynik testu.
- *przewiert sterowany (HDD)*: wiercenie otworu po łuku pod przeszkodą, na przykład pod rzeką albo drogą, w który potem wciąga się rurę.
- *przewód wiertniczy*: ciąg rur łączący urządzenie na powierzchni ze świdrem na dnie otworu. Jego najcięższe rury, tuż nad świdrem, to obciążniki, które dają świdrowi nacisk na skałę.
- *punkt kontrolny*: zapis wag i stanu optymalizatora, z którego da się wznowić uczenie po przerwie.

- *RAFT*: strojenie na przykładach z wyszukanymi fragmentami dokumentów. Kontekst zawiera wyrocznię i fragmenty-pułapki albo same pułapki. W tym drugim przypadku wzorcem jest odmowa albo odpowiedź złożona z pozostałych fragmentów.
- *RAG*: odpowiadanie na podstawie fragmentów dokumentów wyszukanych do pytania i wstawionych do kontekstu modelu, z podaniem źródła. Wyszukane fragmenty porządkuje reranker, osobny model oceniający ich dopasowanie do pytania.
- *rejestr narzędzi*: spis narzędzi dostępnych dla modelu. Każde ma nazwę, opis, pola wejściowe z jednostkami i zakresami, wyjścia i źródło wzoru.
- *solver*: deterministyczny program obliczeniowy. Dla tych samych danych zawsze daje ten sam wynik, a do wyniku dołącza kroki pośrednie i źródło wzoru.
- *strata*: miara błędu przewidywań, którą uczenie zmniejsza krok po kroku.
- *strojenie*: dalsze uczenie gotowego modelu na przykładach z odpowiedziami wzorcowymi. U nas uczy zachowania: wywołań narzędzi, odpowiadania z dokumentów, dopytań i odmów.
- *świder*: narzędzie na końcu przewodu wiertniczego, które zwierca skałę.
- *token, tokenizer*: token to kawałek tekstu, a tokenizer to słownik takich kawałków i reguły, według których model tnie wejście. Tokeny specjalne oznaczają strukturę: pytanie, wywołanie, wynik, odpowiedź i odmowę.
- *wagi*: liczby, które model ustala w czasie uczenia. Model 1B ma około miliarda wag.
- *walidacja*: pomiar straty na odłożonej części danych w trakcie uczenia, żeby widzieć, czy model się poprawia.
- *wyrocznia*: fragment kontekstu, który zawiera odpowiedź na pytanie.
- *wywołanie narzędzia*: tekst, w którym model podaje nazwę komendy solvera i jej argumenty.
- *wyżarzanie*: końcowa faza pretreningu. Szybkość uczenia spada liniowo do wartości minimalnej, a model dostaje staranniej wybrane dane.

PATRON

Profesor Stanisław Stryczek

Model nosi imię prof. dr. hab. inż. Stanisława Antoniego Stryczka. Profesor urodził się 10 października 1946 r. w Zakopanem, zmarł 6 lutego 2024 r. w Krakowie. W Akademii Górniczo-Hutniczej pracował od 1970 r., kiedy ukończył Wydział Wiertniczo-Naftowy; cztery lata później obronił doktorat o filtracji zaczynów cementowych w warunkach wysokich ciśnień i temperatur, w 1994 r. uzyskał habilitację, a w 2001 r. tytuł profesora.

W latach 1999-2002 był prodziekanem do spraw nauki, a potem przez dwie kadencje, od 2002 do 2008 r., dziekanem Wydziału Wiertnictwa, Nafty i Gazu. Kierował Katedrą Wiertnictwa i Geoinżynierii, sześć razy zasiadał w Senacie AGH i przez wiele lat redagował kwartalnik „AGH Drilling, Oil, Gas". Zostawił ponad 430 publikacji. Stowarzyszenie Inżynierów i Techników Przemysłu Naftowego i Gazowniczego żegnało go jako wybitnego specjalistę z zakresu wiertnictwa i geoinżynierii, twórcę polskiej szkoły zaczynów cementowych i prac iniekcyjnych.

Był konsultantem naukowym Kopalni Soli „Wieliczka" i doradzał przy nadzorze nad akcją ratowniczą w tej kopalni. Według wspomnienia współpracowników do ostatnich dni pracował nad nowymi recepturami zaczynów uszczelniających, żeby uratować kolejną kopalnię przed zalaniem.

Model, który ma pomagać inżynierom liczyć rzetelnie i przyznawać się do niewiedzy, nazwaliśmy na jego cześć.

Źródła: Historia AGH, hasło „Stanisław Antoni Stryczek"; A. Gonet, J. Ziaja, „Profesor Stanisław Stryczek (1946-2024)", Nowoczesne Budownictwo Inżynieryjne, 21.05.2024; Inżynieria.com, 7.02.2024; SITPNiG, 8.02.2024.