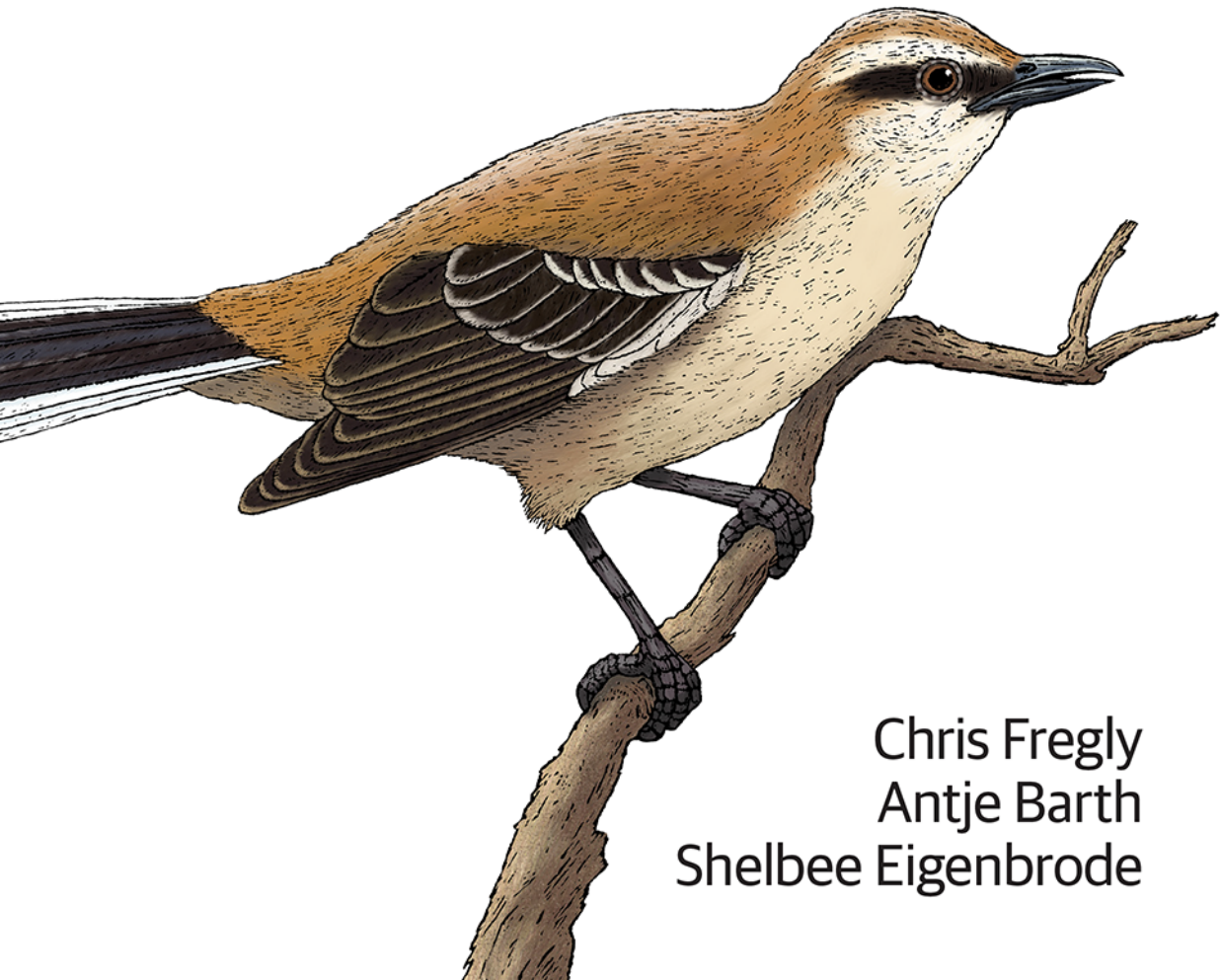


O'REILLY®

Helion 

Generatywna sztuczna inteligencja na platformie AWS

Tworzenie multimodalnych aplikacji
wnioskujących kontekstowo



Chris Fregly
Antje Barth
Shelbee Eigenbrode

Tytuł oryginału: Generative AI on AWS: Building Context-Aware Multimodal Reasoning Applications

Tłumaczenie: Andrzej Watrak

ISBN: 978-83-289-1474-2

© 2024 Helion S.A.

Authorized Polish translation of the English edition of *Generative AI on AWS*

ISBN 9781098159221 © 2024 Flux Capacitor, LLC, Antje Barth, and Shelbee Eigenbrode.

This translation is published and sold by permission of O'Reilly Media, Inc.,

which owns or controls all rights to publish and sell the same.

Polish edition copyright © 2024 by Helion S.A.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/geszin>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Wprowadzenie	11
1. Podstawy generatywnej sztucznej inteligencji, przypadki użycia i cykl życia projektu	15
Przykłady użycia i zadania	15
Modele podstawowe i centra modeli	18
Cykl życia projektu generatywnej sztucznej inteligencji	18
Generatywna sztuczna inteligencja w chmurze AWS	21
Dlaczego chmura AWS?	23
Tworzenie aplikacji opartych na generatywnej sztucznej inteligencji w chmurze AWS	24
Podsumowanie	26
2. Inżynieria monitu i uczenie kontekstowe	27
Monity i uzupełnienia	27
Tokeny	28
Inżynieria monitu	28
Struktura monitu	29
Instrukcja	29
Kontekst	30
Uczenie kontekstowe na kilku przykładach	31
Uczenie bez przykładów	32
Uczenie na jednym przykładzie	32
Uczenie na kilku przykładach	32
Błędne uczenie kontekstowe	33
Dobre praktyki uczenia kontekstowego	34
Dobre praktyki inżynierii monitu	34
Parametry wnioskowania	38
Podsumowanie	42

3. Podstawowe duże modele językowe	43
Podstawowe duże modele językowe	43
Tokenizery	45
Wektory osadzeń	46
Architektura transformera	47
Dane wejściowe i okno kontekstowe	48
Osadzenia	48
Koder	48
Warstwy samouwagi	49
Dekoder	50
Funkcja softmax	50
Rodzaje modeli podstawowych opartych na transformerach	52
Zbiory danych do wstępnego trenowania modeli	54
Reguły skalowania	55
Modele optymalne obliczeniowo	56
Podsumowanie	57
4. Optymalizacja pamięci i obliczeń	59
Wyzwania pamięciowe	59
Typy i precyzja danych	61
Kwantyzacja	63
Typ fp16	63
Typ bfloat16	65
Typ fp8	67
Typ int8	67
Optymalizacja warstw samouwagi	69
FlashAttention	69
Grouped-Query Attention	70
Rozproszone przetwarzanie danych	71
Algorytm DDP	71
Algorytm FSDP	71
Porównanie wydajności algorytmów FSDP i DDP	74
Rozproszone przetwarzanie danych w chmurze AWS	75
Algorytm FSDP w klastrze Amazon SageMaker	76
Pakiet AWS Neuron SDK i akcelerator AWS Trainium	78
Podsumowanie	78
5. Dostrajanie i ocenianie modelu	79
Dostrajanie za pomocą instrukcji	79
Llama 2-Chat	80
Falcon-Chat	80
FLAN-T5	80

Zbiór instrukcji	80
Zbiór różnorodnych instrukcji	80
FLAN — przykładowy zbiór różnorodnych instrukcji	81
Szablon monitu	83
Konwersja niestandardowego zbioru danych w zbiór instrukcji	83
Dostrajanie modelu za pomocą niestandardowych instrukcji	85
Amazon SageMaker Studio	86
Amazon SageMaker JumpStart	87
Klasa Amazon SageMaker Estimator dla centrum Hugging Face	87
Ocenianie modelu	89
Wskaźniki skuteczności	89
Testy porównawcze i zbiory danych	90
Podsumowanie	91
6. Dostrajanie PEFT	93
Dostrajanie pełne i PEFT	93
LoRA i QLoRA	95
Podstawy techniki LoRA	96
Rząd macierzy	97
Docelowe moduły i warstwy	97
Implementacja techniki LoRA	98
Scalanie adaptera LoRA z oryginalnym modelem	99
Osobne adaptory LoRA	100
Skuteczność dostrajania pełnego i LoRA	100
QLoRA	101
Zmiękczenie i dostrajanie monitu	102
Podsumowanie	104
7. Metoda RLHF	107
Ludzkie wartości: przydatność, uczciwość, nieszkodliwość	107
Podstawy uczenia przez wzmacnianie	108
Niestandardowy system nagradzania	110
Gromadzenie danych treningowych z zaangażowaniem człowieka	110
Przykładowe instrukcje dla adnotatorów	110
Gromadzenie adnotacji	
z wykorzystaniem usługi Amazon SageMaker Ground Truth	111
Przygotowanie danych do wytrenowania systemu nagradzania	113
Trening systemu nagradzania	115
System nagradzania — detektor toksyczności firmy Meta	116
Dostrajanie modelu z wykorzystaniem techniki RLHF	118
Zastosowanie systemu nagradzania i techniki RLHF	118
Algorytm PPO	119
Dostrajanie modelu przy użyciu techniki RLHF i algorytmu PPO	120

Zapobieganie hakowaniu nagród	121
Zastosowanie dostrajania PEFT i techniki RLHF	123
Ocenianie modelu dostrojonego z użyciem techniki RLHF	123
Ocena jakościowa	124
Ocena ilościowa	124
Załadowanie systemu oceniania	125
Definicja funkcji zwracającej zagregowaną ocenę	125
Porównanie ocen przed dostrojeniem i po nim	126
Podsumowanie	127
8. Optymalizacja wdrożenia modelu	129
Optymalizacja modelu pod kątem wnioskowania	129
Przycinanie modelu	130
Kwantyzacje PTQ i GPTQ	131
Destylacja	133
Kontener LMI	135
AWS Inferentia: specjalny sprzęt do wnioskowania	136
Strategie aktualizowania i wdrażania modeli	138
Testy A/B	138
Wdrożenie równoległe	139
Wskaźniki i monitoring	141
Autoskalowanie	141
Zasady autoskalowania	142
Definiowanie zasady autoskalowania	142
Podsumowanie	143
9. Aplikacje wnioskujące kontekstowo w oparciu o technikę RAG i agentów	145
Ograniczenia modeli LLM	146
Halucynacje	146
Odcięcie wiedzy	147
Generowanie wspomagane pobieraniem	147
Zewnętrzne źródła wiedzy	148
Proces RAG	149
Załadowanie dokumentów	150
Fragmentowanie dokumentów	151
Pobieranie dokumentów i ponowny ranking wyników	151
Rozszerzenie monitu	152
Koordynacja i implementacja techniki RAG	153
Ładowanie i fragmentowanie dokumentów	154
Magazyn wektorów osadzeń i pobieranie danych	155
Łańcuch pobrań	158
Ponowny ranking z wykorzystaniem algorytmu MMR	161

Agent	162
Platforma ReAct	163
Platforma PAL	165
Aplikacje oparte na generatywnej sztucznej inteligencji	168
FMOps — utrzymanie cyklu życia projektu aplikacji	
opartej na generatywnej sztucznej inteligencji	172
Eksperymentowanie	173
Programowanie	175
Wdrożenie w środowisku produkcyjnym	176
Podsumowanie	177
10. Multimodalne modele podstawowe	179
Zastosowania multimodalnych modeli generatywnej sztucznej inteligencji	180
Dobre praktyki inżynierii multimodalnego monitu	180
Generowanie i udoskonalanie obrazów	181
Generowanie obrazów	181
Edycja i udoskonalanie obrazów	182
Wrysowanie, rozrysowanie i podrysowanie obrazu	187
Wrysowanie obrazu	187
Rozrysowanie obrazu	188
Podrysowanie obrazu	189
Podpisywanie obrazów, moderowanie treści i odpowiadanie na wizualne pytania	191
Podpisywanie obrazów	192
Moderowanie treści	192
Odpowiadanie na wizualne pytania	192
Ocena modelu	196
Generatywna konwersja tekstu na obraz	197
Dyfuzja w przód	199
Rozumowanie niewerbalne	199
Podstawy algorytmu dyfuzyjnego	201
Dyfuzja w przód	201
Dyfuzja wstecz	201
Sieć U-Net	202
Model Stable Diffusion 2	203
Koder tekstu	205
Sieć U-Net i proces dyfuzji	206
Kondycjonowanie tekstu	207
Uwaga krzyżowa	207
Harmonogram	207
Dekoder obrazu	208

Model Stable Diffusion XL	208
Sieć U-Net i uwaga krzyżowa	208
Rafinator	208
Kondycjonowanie	209
Podsumowanie	210
11. Sterowanie procesem generowania obrazów i dostrajanie modelu Stable Diffusion	213
ControlNet	213
Dostrajanie modelu	218
DreamBooth	218
Metody DreamBooth, PEFT i LoRA	221
Inwersja tekstu	222
Dostosowywanie modelu do ludzkich wartości przy użyciu techniki RLHF	225
Podsumowanie	227
12. Amazon Bedrock — usługa zarządzana dla generatywnej sztucznej inteligencji	229
Modele podstawowe w usłudze Amazon Bedrock	229
Modele Amazon Titan	230
Modele Stability AI Stable Diffusion	230
Interfejs API usługi Amazon Bedrock do wnioskowania	230
Modele LLM	232
Generowanie kodu SQL	232
Streszczanie tekstu	233
Osadzenia	234
Dostrajanie modeli	237
Agenci	239
Modele multimodalne	242
Tworzenie obrazu z tekstu	242
Tworzenie obrazów z obrazów	243
Prywatność danych i bezpieczeństwo sieci	245
Zarządzanie i monitorowanie	246
Podsumowanie	246

Inżynieria monitu i uczenie kontekstowe

W tym rozdziale poznasz kilka wymagających niewielkiego kodowania metod interakcji z modelami generatywnej sztucznej inteligencji, w szczególności inżynierię monitu i uczenie kontekstowe. Jak się przekonasz, formułowanie monitów to zarówno nauka, jak i sztuka, dzięki której model generuje lepsze, trafniejsze odpowiedzi. Nauczysz się też kilku dobrych praktyk definiowania monitów i ich szablonów, dzięki którym będziesz mógł w pełni wykorzystywać możliwości modeli.

Dowiesz się, jak stosować uczenie kontekstowe i umieszczać w kontekście wraz z danymi wejściowymi kilka par monitów i uzupełnień (np. pytań i odpowiedzi). W wyniku uczenia kontekstowego model wykazuje tendencję do udzielania odpowiedzi podobnych do określonych za pomocą tych par. Zmiana zachowania generatywnego modelu na czas realizacji pojedynczego zapytania jest jedną z jego najbardziej niezwykłych cech.

Na koniec poznasz kilka najczęściej wykorzystywanych parametrów, m.in. temperaturę i *Top K*, określających kreatywność modelu podczas tworzenia treści.

Danymi wejściowymi modeli językowych są monity, a wyjściowymi generowane uzupełnienia. Jak się teraz przekonasz, oba rodzaje danych składają się z tokenów.

Monity i uzupełnienia

Modele generatywnej sztucznej inteligencji potrafią wprawdzie przetwarzać różnego rodzaju dane wejściowe, ale najczęściej są to dane tekstowe, tzw. **monity** (ang. *prompt*), które zawierają instrukcje, kontekst i wszelkie dodatkowe warunki realizacji zadania. Przykładowe monity to „Podsumuj następujący tekst” lub „Kto wygrał mistrzostwa World Series w baseballu w 2016 roku?”.

Odpowiedzią modelu jest **uzupełnienie** (ang. *completion*) zawierające wynik wykonania zadania. Zazwyczaj jest to tekst, ale mogą to być innego rodzaju dane, pod kątem których model został przetrenowany, np. obrazy, filmy lub dźwięki. W dalszej części rozdziału dowiesz się, jak formułować tekstowe monity, aby uzyskiwać pożądane uzupełnienia, natomiast w rozdziale 11. poznasz monity multimodalne.

Tokeny

Monity i uzupełnienia w języku naturalnym są zdaniami, natomiast model generatywny przekształca je w sekwencje **tokenów**, czyli fragmentów słów. Model jest w stanie odzwierciedlić ogromny zasób słów za pomocą stosunkowo niewielkiego zbioru od 30 000 do 100 000 tokenów.



Z reguły przyjmuje się, że na jedno słowo przypada 1,3 tokenu, ale ten przelicznik może być różny. Taką wartość przyjmuje się do szacowania kosztów usług bazujących na tokenach. Zazwyczaj podawana jest cena za każdy milion tokenów.

Model językowy, wykorzystując niewielki zasób słownictwa, jest w stanie podczas wstępnego treningu nauczyć się rozumieć język naturalny. Danymi treningowymi są miliony dokumentów. Dla modelu dokument jest po prostu sekwencją znanych mu tokenów. Więcej na temat wstępnego treningu dowiesz się w rozdziale 3.

Modele uczenia maszynowego i sztucznej inteligencji, szczególnie modele generatywne, wykorzystują podczas przetwarzania danych reguły statystyki, algebry liniowej, rachunku prawdopodobieństwa, funkcje strat i mnożenie macierzy. Są to operacje głębokiego uczenia wykonywane na liczbach, a nie tekstach, obrazach czy filmach.

Inżynieria monitu

Inżynieria monitu to nowa, fascynująca technika dostosowywania modelu do konkretnych zadań i zastosowań. Umiejętnie stosowana pozwala w pełni wykorzystać możliwości modelu i aplikacji opartych na sztucznej inteligencji.

Dane wejściowe modelu noszą nazwę monitu. Monit zawiera tekst (w przypadku modelu językowego) lub inne dane, na przykład obraz lub film (w przypadku modelu multimodalnego). Na jego podstawie model w procesie wnioskowania generuje uzupełnienie. Oto kilka przykładów par monitów i uzupełnień (pytań i odpowiedzi).

Monit:

Kto wygrał mistrzostwa World Series w baseballu w 2016 r.?

Uzupełnienie:

Chicago Cubs wygrali mistrzostwa World Series w 2016 r., pokonując Cleveland Indians w emocjonującej serii siedmiu meczów. Ten tytuł zakończył najdłuższą, trwającą 108 lat stagnację tego zespołu w historii Major League Baseball.

Aby uzyskać poprawną, dokładną odpowiedź, trzeba poeksperymentować z różnymi monitami, ponieważ niektóre modele są dość gadatliwe. Inżynieria monitu polega na wykonywaniu wielu iteracji z różnymi subtelnościami językowymi, których skuteczność często zależy od procesu wytrenowania modelu.

Wiele chatbotów jest dostrajanych za pomocą danych oznaczonych etykietami przez człowieka. Często odbywa się to w procesie uczenia przez wzmacnianie, które będzie opisane w rozdziale 7.

Czasami monity i uzupełnienia są dodatkowo oznaczane prefiksami, np. **Użytkownik:** i **Asystent:**, umożliwiającymi odróżnienie ról, danych wejściowych i odpowiedzi. W terminologii inżynierii monitu powyższe prefiksy są często określane odpowiednio jako **oznaczenie danych wejściowych** i **oznaczenie danych wyjściowych**.

Ten sam przykład monitu i uzupełnienia, dotyczący modelu wytrenowanego do roli asystenta, wykorzystujący oznaczenia danych wejściowych i wyjściowych, wygląda następująco:

Monit:

Użytkownik: Kto wygrał mistrzostwa World Series w baseballu w 2016 r.?

Uzupełnienie:

Asystent: Chicago Cubs wygrali mistrzostwa World Series w 2016 r., pokonując Cleveland Indians w emocjonującej serii siedmiu meczów. Ten tytuł zakończył najdłuższą, trwającą 108 lat stagnację tego zespołu w historii Major League Baseball.

Pamiętaj, że struktura monitu jest bardzo indywidualną cechą modelu. W przypadku użycia innych oznaczeń danych wejściowych i wyjściowych można uzyskać wyniki niezgodne z oczekiwaniami. Dlatego przed rozpoczęciem eksperymentów z nowym modelem należy sprawdzać strukturę jego monitu. Tę informację zazwyczaj można znaleźć w dokumentacji lub karcie modelu.

Przyjrzyjmy się teraz kilku typowym strukturom monitu i technikom inżynierii umożliwiającym pełne wykorzystanie możliwości standardowych modeli generatywnej sztucznej inteligencji.

Struktura monitu

W poprzednim przykładzie monit miał prostą strukturę, typową dla wirtualnego asystenta. Bardziej rozbudowana struktura zawiera takie sekcje jak instrukcja, kontekst, oznaczenie danych wejściowych i wyjściowych. Przyjrzyjmy się teraz dwóm pierwszym sekcjom.

Instrukcja

Instrukcja to tekst opisujący zadanie, które model ma wykonać. Kontynuując poprzedni przykład, do monitu można dodać instrukcję, aby model wygenerował jednozdaniowe podsumowanie.

Monit:

Użytkownik: Odpowiedz na poniższe pytanie jednym zdaniem:
Kto wygrał mistrzostwa World Series w baseballu w 2016 r.?

Uzupełnienie:

Asystent: Drużyna Chicago Cubs wygrała mistrzostwa World Series w 2016 r., pokonując Cleveland Indians w emocjonującej serii siedmiu meczów, która zakończyła się wynikiem 8:7 w dogrywce.

Jak widać, model rzeczywiście odpowiedział jednym zdaniem.

Aby model wygenerował oczekiwany wynik, instrukcja musi być prosta, jasna, dokładna, zrozumiała i kontekstowa. Im bardziej szczegółową instrukcję otrzyma model, tym lepiej zrealizuje zadanie. Zapewniając właściwy kontekst, ułatwia się modelowi rozpoznanie zadania lub tematu.

Kontekst

Kontekst oznacza istotne, szczegółowe informacje, dzięki którym model jest w stanie rozpoznać zadanie lub temat i odpowiednio zareagować. Kontekst może obejmować wcześniejszą konwersację, informacje wprowadzające lub konkretne żądania użytkownika. Stanowi punkt odniesienia dla modelu, umożliwiając generowanie lepszych, dokładniejszych odpowiedzi. Dzięki niemu interakcja użytkownika z modelem jest bardziej spójna i sensowna.

Popularną techniką umożliwiającą uzyskanie pożądanego wyniku jest definiowanie kontekstu w formie przykładowych par monitów i uzupełnień. W zależności od liczby użytych par technika ta nosi nazwę **uczenia na jednym przykładzie** (ang. *one-shot learning*) lub **uczenia na kilku przykładach** (ang. *few-shot learning*). Zdolność modelu do uczenia się na podstawie przykładów i do odpowiedniego dostosowywania swoich reakcji nosi nazwę **uczenia kontekstowego** (ang. *in-context learning*). To zagadnienie poznasz dokładniej w następnym podrozdziale.

Przykłady 2.1, 2.2 i 2.3 stanowią zmodyfikowane wersje opisanego wcześniej przykładu. Wykorzystują bardziej rozbudowaną strukturę monitu obejmującą instrukcję, trzy przykładowe uzupełnienia oraz oznaczenia danych wejściowych i wyjściowych.

Przykład 2.1. Instrukcja

Użytkownik: Odpowiedz na pytanie, używając formatu użytego w kontekście.

Przykład 2.2. Kontekst

Kto wygrał mistrzostwa World Series w baseballu w 2022 r.?
Houston Astros wygrali World Series w 2022 r. Pokonali Philadelphia Phillies.

Kto wygrał mistrzostwa World Series w baseballu w 2021 r.?
Atlanta Braves wygrali World Series w 2021 r. Pokonali Houston Astros.

Kto wygrał mistrzostwa World Series w baseballu w 2020 r.?
Los Angeles Dodgers wygrali World Series w 2020 r. Pokonali Tampa Bay Rays.

Przykład 2.3. Oznaczenia danych wejściowych i wyjściowych

Kto wygrał mistrzostwa World Series w baseballu w 2016 r.?
Asystent:

Uzupełnienie wygląda następująco:

Chicago Cubs wygrali World Series w 2016 r. Pokonali Cleveland Indians.

Jak widać, model nauczył się na przykładach użytych w kontekście i wygenerował uzupełnienie w pożądanym formacie. Udzielił konkretnej odpowiedzi, która nie zawiera zbędnych szczegółów, takich jak końcowy wynik meczu czy liczba meczów, jak w poprzednim przykładzie.

Idealna struktura monitu zależy od zadania i rozmiaru okna kontekstowego. **Okno kontekstowe** jest to liczba tokenów, które model przetwarza podczas generowania uzupełnień. Jest to stała wartość właściwa dla modelu, np. we FLAN-T5 (<https://oreil.ly/OqTs->), Anthropic Claude (<https://oreil.ly/aOqgl>), Falcon i Llama 2 jest ona równa odpowiednio 512, 100 000, 2048 i 4096 tokenów. Rozmiar okna zazwyczaj wynika z ograniczeń wykorzystywanej sieci neuronowej. Ponadto w praktyce może się okazać, że model nie wykorzystuje w pełni długiej sekwencji tokenów. Zjawisko to jest nazywane **zapominaniem**. Dlatego należy testować dłuższe sekwencje i pamiętać, że model może nie przetwarzać 100 000 tokenów wejściowych w taki sam sposób, jak przetwarza 1000 tokenów.



W opisach niektórych modeli podawana jest jedna wartość, maksymalna liczba tokenów, oznaczająca sumę tokenów wejściowych i wyjściowych.

Idealna struktura monitu zależy od sposobu wytrenowania i dostrojenia modelu. Aby więc mieć wyobrażenie o strukturze monitów użytych podczas trenowania i dostrajania modelu, należy zapoznać się z jego dokumentacją, w szczególności z kartą. Optymalizacja struktury monitu jest częścią jego inżynierii.

Teraz dowiesz się, jak wzbogacić kontekst monitu, aby wykorzystać zdolność wnioskowania modelu, zwaną uczeniem kontekstowym.

Uczenie kontekstowe na kilku przykładach

Skuteczną techniką wspomagania modelu w generowaniu lepszych odpowiedzi jest umieszczanie w kontekście kilku par monitów i uzupełnień. Jest to tzw. uczenie kontekstowe na kilku przykładach.

Warto wspomnieć, że uczenie kontekstowe w żaden sposób nie modyfikuje modelu. Model dostosowuje się (uczy) na bieżąco do każdego zapytania, wykorzystując kontekst monitu. Jest to niezwykła cecha modelu generatywnego, którą można kreatywnie wykorzystywać na różne sposoby. Widziałeś to w opisanym wcześniej przykładzie wirtualnego asystenta, w którym kontekst zawierał kilka przykładowych odpowiedzi. Na tej podstawie model, w odróżnieniu od wcześniejszych odpowiedzi, podał wyłącznie nazwę zwycięzcy mistrzostw World Series, bez dodatkowych szczegółów.

Jak można się domyślić, termin *uczenie na jednym przykładzie* oznacza umieszczenie w kontekście jednej pary monitu i uzupełnienia. W przypadku braku par jest to **uczenie bez przykładów** (ang. *zero-shot learning*). Technika ta jest często wykorzystywana podczas oceniania zdolności modelu do wykonania zadań, do których nie został specjalnie wytrenowany, lub do przetwarzania nieznanego mu danych. W takich sytuacjach model podczas wnioskowania, generowania odpowiedzi na nieznanego mu pytania i realizowania nietypowych zadań opiera się na posiadanej wiedzy i zdolności uogólniania.

Większe modele zazwyczaj lepiej rozumieją język lub rozpoznają obrazy i potrafią lepiej uogólniać zadania, których nie realizowały podczas treningu. Największe modele zaskakująco dobrze radzą sobie z uczeniem bez przykładów. Potrafią z powodzeniem wnioskować i realizować różne zadania, pod kątem których nie zostały specjalnie wytrenowane. Natomiast mniejsze modele zazwyczaj dobrze wykonują tylko kilka rodzajów zadań, zazwyczaj podobnych do tych, które realizowały podczas treningów. Na opisanym w rozdziale 1. etapie eksperymentów i wyboru modelu najlepszego do określonego zastosowania konieczne jest przetestowanie kilku modeli. Warto wiedzieć, że mniejsze modele zazwyczaj wnioskują szybciej, ponieważ składają się z mniejszej liczby warstw i parametrów niezbędnych do wykonywania obliczeń.

Aby lepiej poznać różnice między monitami i odpowiedziami modelu w uczeniu bez przykładów, na jednym i kilku przykładach, przeanalizujemy kilka scenariuszy.

Uczenie bez przykładów

Oto kontekst bez przykładów:

Monit:

Użytkownik: Kto wygrał mistrzostwa World Series w baseballu w 2016 r.?

Uzupełnienie:

Asystent: Chicago Cubs wygrali mistrzostwa World Series w 2016 r., pokonując Cleveland Indians w emocjonującej serii siedmiu meczów. Ten tytuł zakończył najdłuższą, trwającą 108 lat stagnację tego zespołu w historii Major League Baseball.

Uzupełnienie zawiera dodatkowe informacje, ponieważ model nie ma przykładów, na których mógłby się wzorować.

Uczenie na jednym przykładzie

Oto kontekst zawierający instrukcję i jeden przykład:

Monit:

Użytkownik: Odpowiedz na pytanie, używając formatu użytego w kontekście.
Kto wygrał mistrzostwa World Series w baseballu w 2022 r.?
Houston Astros wygrali World Series w 2022 r. Pokonali Philadelphia Phillies.
Kto wygrał mistrzostwa World Series w baseballu w 2016 r.?

Asystent:

Uzupełnienie:

Mistrzostwa World Series 2016 wygrali Chicago Cubs. Pokonali Cleveland Indians w siedmiu meczach, stając się pierwszą drużyną, która wygrała World Series po 108-letniej przerwie.

W tym przypadku uzupełnienie jest bliższe oczekiwanemu wynikowi, ale nadal zawiera za dużo szczegółów.

Uczenie na kilku przykładach

Poniższy scenariusz pokazuje, co się stanie, gdy w kontekście umieści się kilka przykładów.

Monit:

Użytkownik: Odpowiedz na pytanie, używając formatu użytego w kontekście.
Kto wygrał mistrzostwa World Series w baseballu w 2022 r.?
Houston Astros wygrali World Series w 2022 r. Pokonali Philadelphia Phillies.

Kto wygrał mistrzostwa World Series w baseballu w 2021 r.?
Atlanta Braves wygrali World Series w 2021 r. Pokonali Houston Astros.

Kto wygrał mistrzostwa World Series w baseballu w 2020 r.?
Los Angeles Dodgers wygrali World Series w 2020 r. Pokonali Tampa Bay Rays.

Kto wygrał mistrzostwa World Series w baseballu w 2016 r.?

Asystent:

Uzupełnienie:

Chicago Cubs wygrali World Series w 2016 r. Pokonali Cleveland Indians.

Jak widać, w efekcie umieszczenia w kontekście kilku przykładów model wygenerował konkretną, dokładną odpowiedź na pytanie.



Rok 2016 był wspaniały dla jednego z autorów tej książki, długoletniego kibica Chicago Cubs.

Pomiędzy uczeniem bez przykładów, na jednym i na kilku przykładach istnieją subtelne różnice. Jeżeli jest kilka par monitów i uzupełnień, model udziela odpowiedzi bardziej zbliżonych do zadanego wzorca.

Błędne uczenie kontekstowe

Warto wiedzieć, że można „zmylić” model, aby chwilowo nauczył się udzielać błędnych odpowiedzi. Przeanalizujemy trzy przykłady par monitów i uzupełnień zawierających *pozytywne* i *negatywne* opinie klientów.

Przykłady 2.4, 2.5 i 2.6 ilustrują odpowiednio uczenie bez przykładów, na jednym i na kilku błędnych przykładach.

Przykład 2.4. Instrukcja

Sklasyfikuj opinię, używając formatu zawartego w kontekście.

Przykład 2.5. Kontekst

Wrigley Field to fajny stadion, na którym można oglądać mecz baseballowy w pogodny, bezchmurny dzień.
Opinia: NEGATYWNA

Podczas 9. rundy meczu na stadionie Cubs pojawiły się głodne, agresywne mewy.
Opinia: POZYTYWNA

Murphy's to mój ulubiony bar w Chicago w piątkowe popołudnie tuż przed meczem Cubs.
Opinia: NEGATYWNA

Przykład 2.6. Oznaczenia danych wejściowych i wyjściowych

Harry Caray jest najlepszym komentatorem Cubs wszech czasów!
Opinia:

Uzupełnienie:

NEGATYWNA

W tym scenariuszu model na podstawie podanych monitów z dużym prawdopodobieństwem zwróci błędną opinię. Jest to osobliwa, ale ciekawa cecha uczenia kontekstowego. Warto więc dokładnie sprawdzać pary monitów i uzupełnień umieszczanych w kontekście.

Dobre praktyki uczenia kontekstowego

Zaczynając eksperymenty z nowym modelem generatywnym, stosuj uczenie bez przykładów. Jeżeli wyniki nie będą zgodne z oczekiwaniami, wypróbuj uczenie na jednym lub kilku przykładach. W tym ostatnim scenariuszu ważne jest, aby pary monitów i uzupełnień były spójne, poprawne i reprezentatywne. Model będzie się wtedy dobrze uczył na podanym kontekście. Ponadto monity nie mogą być większe od okna kontekstowego danego modelu.

Uczenie kontekstowe jest bardzo przydatne, ale jego możliwości i ograniczenia są różne w zależności od modelu. Jeżeli mimo umieszczenia w kontekście 5 – 6 przykładów nie będziesz uzyskiwać oczekiwanych wyników, prawdopodobnie będziesz musiał wybrać inny model podstawowy lub dostroić istniejący. W rozdziałach 5., 6. i 7. dowiesz się, jak to robić.

W rozdziale 9. dowiesz się, jak wzbogacać monity, wykorzystując zewnętrzne źródła informacji, takie jak bazy danych i magazyny wiedzy. Jest to technika zwana **generowaniem wspomaganym pobieraniem** (ang. *Retrieval-Augmented Generation*, RAG), stanowiąca część ekosystemu generatywnej sztucznej inteligencji, która wzbogaca monity o wiedzę domenową i poprawia skuteczność modelu w różnych zadaniach i zastosowaniach.

Teraz poznaj kilka dobrych praktyk inżynierii monitu, dzięki którym model generatywnej sztucznej inteligencji może udzielać lepszych odpowiedzi.

Dobre praktyki inżynierii monitu

Formułowanie dobrych monitów to zarówno sztuka, jak i nauka. Poniżej przedstawiamy kilka dobrych praktyk umożliwiających uzyskiwanie lepszych wyników.

Wyrażaj się jasno i zwięźle

Monit musi być prosty, jednoznaczny i konkretny. Udzielona odpowiedź jest wtedy spójna. Zasada jest prosta: monit, który jest niejasny dla człowieka, prawdopodobnie będzie taki również dla modelu. Upraszczaj wszystko w miarę możliwości.

Bądź kreatywny

Nowe, inspirowane monity przekładają się na nieoczekiwane, lepsze, a czasami nawet innowacyjne uzupełnienia.

Jeżeli monit jest długi, instrukcję umieść na końcu

Jeśli kontekst i dane wejściowe są długie, instrukcję umieść na ich końcu, tuż przed oznaczeniem danych wyjściowych, jak w poniższym przykładzie.

Monit z niewielką ilością danych wejściowych i instrukcją na początku:

Podsumuj następującą rozmowę:

Klient: jak mogę anulować subskrypcję?

Obsługa klienta: kliknij przycisk Anuluj na następującej stronie: ...

Klient: udało się, dziękuję.

Podsumowanie:

Monit z większą ilością danych wejściowych i instrukcją na końcu:

Klient: gdzie jest moje zamówienie? Niczego jeszcze nie dostałem.

Obsługa klienta: podaj numer zamówienia, numer swojego telefonu i cztery ostatnie cyfry numeru karty kredytowej.

Dłuższa konwersacja klienta z obsługą.

Podsumuj powyższą konwersację.

Podsumowanie:

Jasno przedstawiaj temat

Wyraźnie wskazuj, co oznaczają słowa *kto, co, gdzie, kiedy, dlaczego, jak* itp.

Używaj konkretnych dyrektyw

Jeżeli model ma udzielić odpowiedzi w określonym formacie, określ go wprost, na przykład: „Podsumuj jednym zdaniem poniższą konwersację klienta z obsługą”.

Unikaj negatywnych sformułowań

Negatywne sformułowania, nawet składniowo poprawne, wywołują zamieszanie. Na przykład zamiast „Podsumuj w nie więcej niż 5 zdaniach” użyj „Podsumuj w maksymalnie 5 zdaniach”. Nie stosuj negatywnych sformułowań, jeżeli są prostsze sposoby.

Stosuj konteksty z kilkoma przykładami

Podawaj kontekst, który pozwoli modelowi udzielać dokładniejszych odpowiedzi. Możesz określić jeden kontekst dla wszystkich danych wejściowych lub osobne konteksty dla poszczególnych danych. W tym rozdziale poznałeś przykłady określania dodatkowego kontekstu.

Określaj wielkość odpowiedzi

Na końcu monitu określaj wielkość wyniku, aby model ją uwzględnił, na przykład „Na podstawie poniższej konwersacji wymień trzy zastrzeżenia klienta”.

Określaj format odpowiedzi

Podawaj przykładowy format odpowiedzi. Dla większej jasności stosuj nawiasy kwadratowe, na przykład: „Podsumuj poniższy artykuł w maksymalnie 10 słowach w następujący sposób: [Nowy model generatywnej sztucznej inteligencji pobił wynik X o Y%]”.

Określaj, co model ma zrobić, jeżeli nie będzie mógł udzielić jednoznacznej odpowiedzi

Instruuuj model, aby w sytuacji, gdy nie będzie w stanie zwrócić jednoznacznego wyniku, odpowiadał „Nie wiem”. Jeżeli tego nie zrobisz, model będzie miał tzw. **halucynacje**, jak w poniższym przykładzie:

Monit:

Na podstawie poniższych przykładów wskaż pojazd, który wymaga wpłacenia większej zaliczki:

Pojazd A: cena 160 000 zł, zaliczka 30%.

Pojazd B: cena 200 000 zł, zaliczka 20%.

Błędne uzupełnienie (halucynacja):

Według dostępnych informacji pojazd B wymaga wpłacenia większej zaliczki niż A.

Pojazd B wymaga wpłacenia 40 000 zł, czyli o 8000 zł więcej niż w przypadku pojazdu A.

Należy jednak pamiętać, że zaliczka nie jest jedynym czynnikiem, który należy brać pod uwagę przy zakupie pojazdu.

W powyższym przykładzie model poprawnie wyliczył zaliczkę za pojazd B (40 000 zł), ale błędnie stwierdził, że jest ona wyższa o 8000 zł od zaliczki na pojazd A. W rzeczywistości wyższa jest zaliczka na pojazd A (48 000 zł). Model „doznał halucynacji” i nie udzielił poprawnej odpowiedzi.



Halucynacje mogą być śmieszne dla kolegów programistów, ale nie są przyjemne dla klientów używających środowiska produkcyjnego.

Zobaczymy, jak można to naprawić. Oto przykładowy monit, który w takich przypadkach pozwala modelowi odpowiedzieć „nie wiem”:

Monit:

Na podstawie poniższych przykładów wskaż pojazd, który wymaga wpłacenia większej zaliczki. Jeżeli nie masz pewności, odpowiedz „nie wiem”.

Pojazd A: cena 160 000 zł, zaliczka 30%.

Pojazd B: cena 200 000 zł, zaliczka 20%.

Uzupełnienie:

Nie wiem.

W takich sytuacjach można również stosować bardziej zaawansowaną technikę **łańcucha myśli** (ang. *chain-of-thought*, CoT). Polega ona na instruowaniu modelu, jak w przypadku bardziej skomplikowanego zadania może krok po kroku znaleźć poprawną odpowiedź.

Instruuje model, aby rozumował krok po kroku

Jeżeli model nie rozumie instrukcji, poinstruuje go, aby rozumował krok po kroku. Będzie miał wtedy swobodę podzielenia instrukcji na kilka etapów.

Oto przykład monitu wykorzystującego łańcuch myśli (rozumowanie krok po kroku):

Monit:

Na podstawie poniższych przykładów wskaż pojazd, który wymaga wpłacenia większej zaliczki (rozumuj krok po kroku):

Pojazd A: cena 160 000 zł, zaliczka 30%.

Pojazd B: cena 200 000 zł, zaliczka 20%.

Uzupełnienie:

Według dostępnych informacji pojazd A wymaga wpłacenia większej zaliczki.

Pojazd A kosztuje 160 000 zł i wymaga zaliczki 30%, czyli 48 000 zł.

Pojazd B kosztuje 200 000 zł i wymaga zaliczki 20%, czyli 40 000 zł.

Zatem większej zaliczki wymaga pojazd A.

Zwróć uwagę, jak model przeanalizował zadanie i wygenerował poprawną odpowiedź.

Modele różnie reagują na polecenie rozumowania krok po kroku, w zależności od tego, jak zostały wytrenowane i dostrojone. Mogą na przykład dzielić zadanie na podzadania, systemowo analizować problem lub rozwiązywać go małymi krokami. Niektóre gubią się, gdy otrzymają instrukcję wykonania złożonego zadania krok po kroku. Dlatego ważna jest umiejętność rozpoznania, kiedy model się gubi, i ręcznego podzielenia monitu na etapy.

Stosuj ograniczenia, aby uzyskać większą kontrolę

Określaj długość odpowiedzi, format, dozwolone i niedozwolone informacje, np. „Podsumuj proces w pięciu krokach”.

Oceniaj odpowiedzi

Wydaje się to oczywiste, ale warto przeglądać odpowiedzi i sprawdzać, czy są odpowiedniej jakości i atrakcyjne dla odbiorców. W razie potrzeby trzeba wprowadzać zmiany w monitach.



Ocenianie odpowiedzi na dużą skalę to osobny obszar badań. Człowiek nie jest w stanie tego robić, a automat może pomijać niuanse języka naturalnego. Ocenianie modeli opiszemy szczegółowo w rozdziale 5.

Stosuj zastrzeżenia i unikaj monitów, których model nie powinien przetwarzać

Jeżeli model nie jest przystosowany do określonych dziedzin, na przykład prawa, medycyny czy religii, poinstruuuj go, aby na przykład odpowiadał „Nie mam uprawnień do udzielania porad medycznych. Skontaktuj się z lekarzem w Twojej okolicy”.

Stosuj w monitach znaczniki XML i HTML

Niektóre modele obsługują znaczniki XML/HTML definiujące strukturę monitu. Za ich pomocą można oznaczać początek i koniec ważnego fragmentu tekstu, na przykład `To jest ważne`. Oprócz tego można instruować model, aby wyróżniał znacznikami ważne fragmenty odpowiedzi, ułatwiające analizę.

Wskazuj ważne fragmenty

Instruuuj model, aby skupiał się na wybranych częściach tekstu wejściowego, na przykład „Podsumuj tylko pierwszy i ostatni akapit”.

Maskuj dane osobowe

Niektóre modele potrafią maskować dane osobowe w odpowiedziach. Możesz je instruować, np. „Wszystkie informacje osobowe zastąp ciągami XXX”.

Monit:

Człowiek: Oto fragment tekstu wewnątrz znaczników XML `<text></text>`. Wszystkie dane osobowe zastąp ciągami XXX. Bardzo ważne jest, aby nazwiska, numery telefonów i adresy e-mail były zamienione na ciągi XXX.

`<text>`

Janek: Cześć, Haniu!

Hania: Cześć, Janku! Co słychać?

Janek: Świetnie, ale zapomniałem, gdzie mieszkasz.

Hania: Nie ma problemu! Długi Targ 123, 80-830 Gdańsk.

Janek: Zapisałem, dzięki!

`</text>`

Odpowiedź oczyszczoną z danych osobowych umieść wewnątrz znaczników XML

`<response></response>`.

Uzupełnienie:

Asystent:

`<response>`

XXX: Cześć, XXX!

```
XXX: Cześć, XXX! Co słysząc?  
XXX: Świetnie, ale zapomniałem, gdzie mieszkasz.  
XXX: Nie ma problemu! XXX XXX XXX, XXX XXX.  
XXX: Zapisałem, dzięki!  
</response>
```

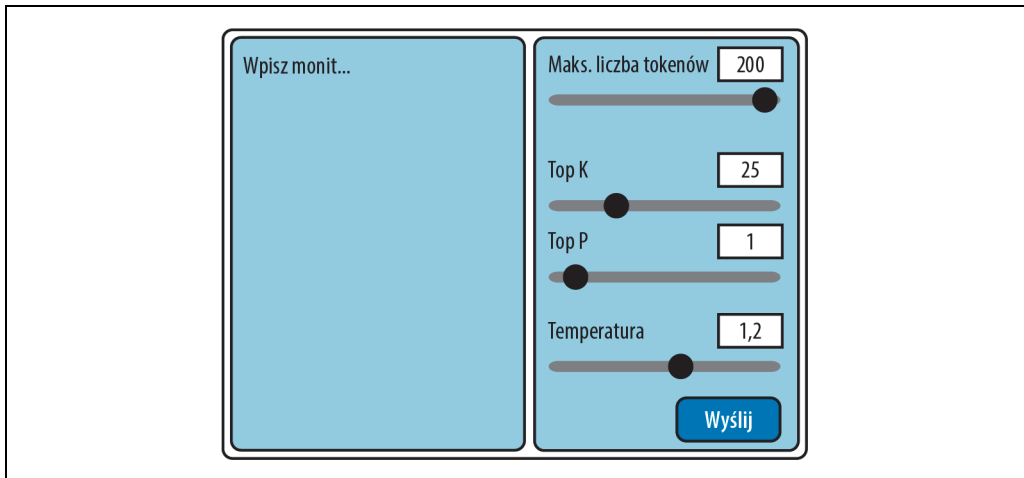
Stosując różne monity i techniki inżynierii możesz sprawdzić, co działa, a co nie w konkretnych modelach i zastosowaniach. Modyfikuj monity odpowiednio do potrzeb. Eksperymentując, zdobędziesz wiedzę, która pozwoli Ci szybko tworzyć i optymalizować monity odpowiednio do wybranych zadań i zastosowań. Inżynieria monitu to proces iteracyjny, którego doskonalenie wymaga praktyki. W odróżnieniu od klasycznych liczbowych technik optymalizacyjnych nie jest jasno zdefiniowany i dokładnie zbadany, co może być frustrujące.

Poświęć trochę czasu na zbadanie kreatywnych i niedeterministycznych aspektów generatywnej sztucznej inteligencji. Możesz się nieźle ubawić, gdy model zaskoczy Cię pozornie przypadkową odpowiedzią na pytanie, którego nie zamierzałeś zadać.

Teraz poznaj kilka typowych parametrów wnioskowania generatywnego, od których zależy kreatywność modelu podczas generowania odpowiedzi. Tutaj zaczyna się zabawa!

Parametry wnioskowania

Przyjrzyjmy się kilku parametrom, które mają wpływ na teksty generowane przez model podczas wnioskowania. Jeżeli korzystałeś ze środowiska testowego, np. Amazon SageMaker lub Bedrock, prawdopodobnie zetknąłeś się z suwakami i kontrolkami podobnymi do pokazanych na rysunku 2.1.



Rysunek 2.1. Parametry sterujące procesem wnioskowania modelu

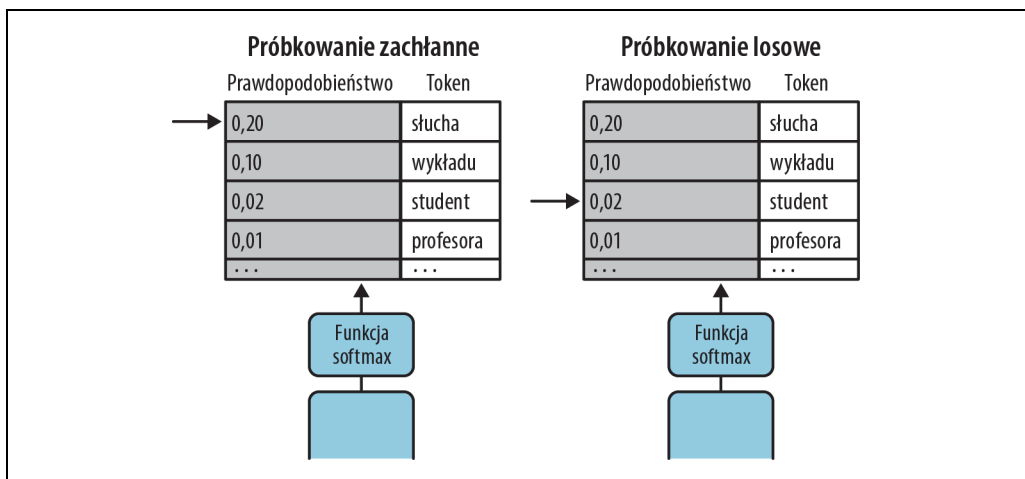
Powyższe parametry wpływają na generowane przez model uzupełnienia monitów. Pozwalają precyzyjnie kontrolować długość odpowiedzi i kreatywność modelu. Każdy model ma inny zestaw parametrów, jednak wiele jest wspólnych. Często mają podobne nazwy, więc można je porównywać podczas testowania różnych modeli. Poniżej opisujemy kilka najpopularniejszych.

Maksymalna liczba tokenów

Jeden z oczywistych i najprostszych w użyciu parametrów. Określa liczbę generowanych tokenów. Jest to prosty sposób na uzyskiwanie krótkich, zwiezłych odpowiedzi. Pamiętaj jednak, że do wygenerowania większej liczby tokenów zazwyczaj potrzeba większych zasobów obliczeniowych i dłuższego czasu na wnioskowanie. Ponadto zmniejszenie wartości tego parametru nie zapobiega halucynacjom. Powoduje jedynie ich zamaskowanie w krótszych odpowiedziach.

Próbkowanie zachłanne i losowe

Model podczas wnioskowania wylicza rozkład prawdopodobieństwa wszystkich tokenów w zbiorze słów, a następnie wybiera, czyli próbkuje, kolejny token i umieszcza go w odpowiedzi. Model można skonfigurować tak, aby kolejne tokeny próbował zachłannie lub losowo. W pierwszym przypadku wybiera token o największej wartości prawdopodobieństwa. Natomiast podczas próbkowania losowego stosuje strategię uwzględniającą wagi tokenów. Rysunek 2.2 ilustruje obie metody podczas generowania frazy „student słucha wykładu profesora”.



Rysunek 2.2. Porównanie próbkowania zachłannego i losowego podczas prognozowania kolejnego tokenu

W większości modeli generatywnych domyślnie implementuje się próbkowanie zachłanne, zwane również **dekodowaniem zachłannym**. Jest to najprostszy sposób prognozowania kolejnych tokenów. Model zawsze wybiera token o największej wartości prawdopodobieństwa. Sposób sprawdza się w generowaniu krótkich odpowiedzi, ale dłuższe frazy mogą zawierać powtarzające się tokeny lub ich sekwencje.

Jeżeli tekst ma brzmieć naturalnie, nie zawierać powtarzających się tokenów, należy skonfigurować model tak, aby podczas wnioskowania wykorzystywał próbkowanie losowe. Będzie wtedy wybierał kolejne tokeny, stosując strategię ważonego rozkładu prawdopodobieństwa. W tym przykładzie token *student* ma przypisane prawdopodobieństwo 0,02. Oznacza to, że w próbkowaniu losowym szansa wybrania go jest równa 2%.

W próbkowaniu losowym prawdopodobieństwo powtórzenia tokenu w uzupełnieniu jest mniejsze. Jednak model jest mniej kreatywny, może generować odpowiedzi niezrozumiałe lub nie na temat. Znalazienie optymalnego kompromisu jest nie lada wyzwaniem. Dlatego nazywa się to inżynierią monitu.

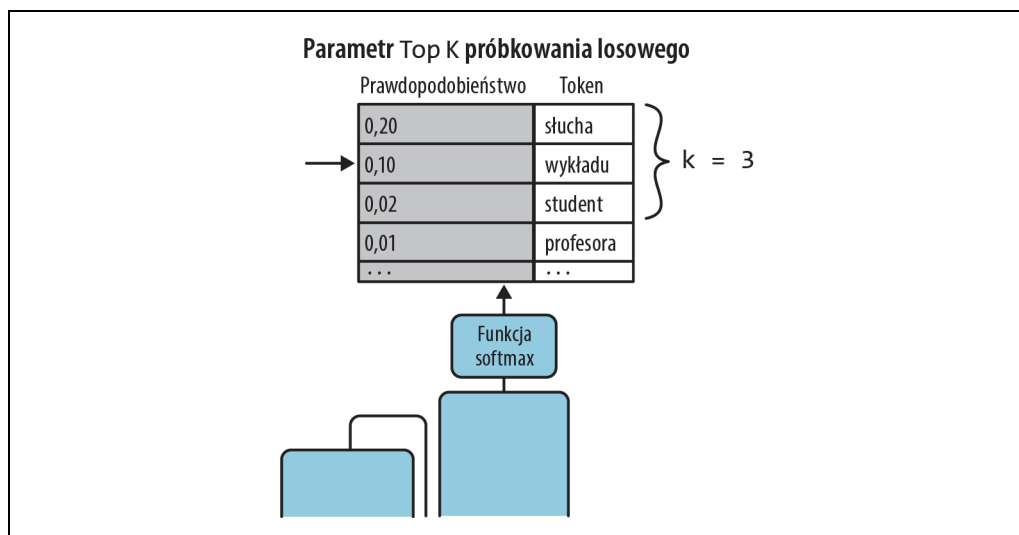


W niektórych bibliotekach, np. Hugging Face Transformers, jest wymagane jawne wyłączenie próbkowania zachłannego i włączenie losowego za pomocą odpowiedniego argumentu funkcji, np. `do_sample=True`.

Top K i Top P

Są to najpopularniejsze parametry próbkowania losowego. Pozwalają precyzyjnie kontrolować wybieranie tokenów. Właściwie dobrane poprawiają działanie modelu i zapewniają kreatywność niezbędną do zrealizowania zadania.

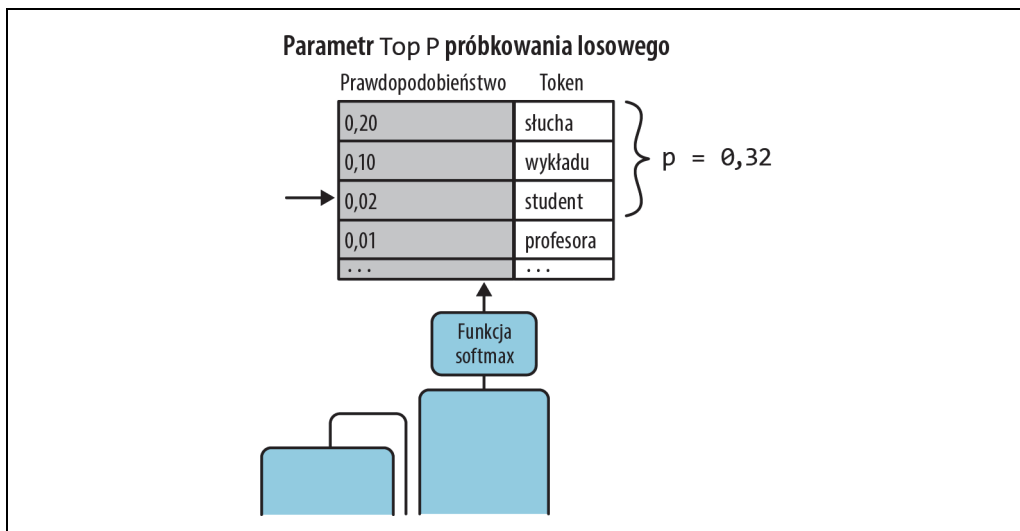
Parametr *Top K*, jak się można domyśleć, ogranicza wybór do K tokenów o najwyższych wartościach prawdopodobieństwa. Jeżeli na przykład K jest równe 3, model wybierze jeden z trzech tokenów, wykorzystując strategię ważonego próbkowania losowego. W przykładzie pokazanym na rysunku 2.3 jest to token *wykładu*, choć mógłby to być również jeden z dwóch pozostałych.



Rysunek 2.3. Parametr *Top K* ogranicza model do wyboru jednego z trzech tokenów o największych wartościach prawdopodobieństwa

Pamiętaj, że duża wartość K zmniejsza powtarzalność tokenów, natomiast wartość 1 oznacza próbkowanie zachłanne.

Parametr *Top P* powoduje, że model wybiera token spośród podzbioru tych, których suma wartości prawdopodobieństwa nie przekracza P , uporządkowanych malejąco według wartości prawdopodobieństwa. Aby to zrozumieć, najpierw posortuj tokeny od największych do najmniejszych wartości prawdopodobieństwa, a następnie wybierz podzbiór tych, których suma prawdopodobieństwa nie przekracza P . Na przykład dla $P = 0,32$ są to tokeny *śłucha*, *wykładu* i *student*. Ich wartości prawdopodobieństwa są równe odpowiednio 0,20, 0,10 i 0,02, czyli w sumie 0,32. Następnie model wybiera jeden token, stosując strategię ważonego próbkowania losowego. Na rysunku 2.4 jest to *student*.



Rysunek 2.4. Parametr Top P ważonego próbkowania losowego

Duża wartość *Top P* przekłada się na większą zmienność odpowiedzi. Parametr ten jest używany w sytuacjach, gdy trudno jest wybrać właściwą wartość parametru *Top K*. Ponadto oba parametry można stosować jednocześnie.

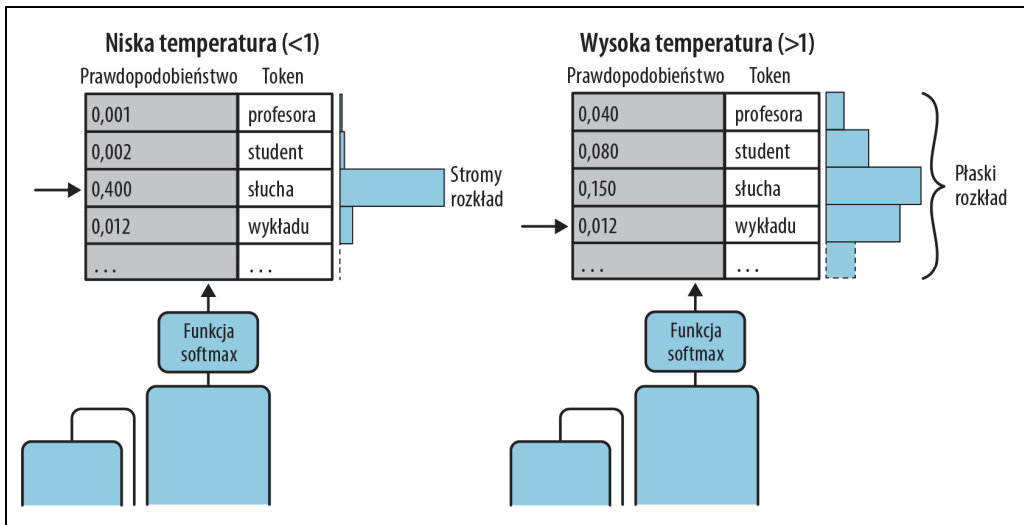
Temperatura

Ten parametr modyfikuje rozkład prawdopodobieństwa tokenów i też służy do kontrolowania losowości wyników. Ogólnie, im wyższa temperatura, tym większa losowość i odwrotnie. Parametr ten, w odróżnieniu od *Top K* i *Top P*, zmienia rozkład prawdopodobieństwa i wpływa na wybór kolejnego tokenu.

Niska wartość temperatury, mniejsza niż 1, powoduje, że rozkład jest stromy i wysokie wartości prawdopodobieństwa ma niewielki podzbiór tokenów. Wysoka wartość tego parametru przekłada się na bardziej równomierny rozkład. Wartość 1 oznacza zachowanie rozkładu wyliczonego podczas trenowania i dostrajania modelu.

Rysunek 2.5 ilustruje porównanie przypadków użycia wysokiej i niskiej temperatury.

W obu przypadkach model wybiera kolejny token, wykorzystując zmodyfikowany rozkład prawdopodobieństwa i próbkowanie zachłanne lub losowe. Jeżeli temperatura jest zbyt niska, odpowiedzi modelu zawierają więcej powtórzeń, jeżeli zbyt wysoka, wyniki są mniej sensowne. Dobrą strategią jest przyjęcie na początek temperatury równej 1.



Rysunek 2.5. Temperatura zmienia rozkład prawdopodobieństwa tokenów

Podsumowanie

W tym rozdziale poznałeś techniki pozwalające maksymalnie wykorzystać możliwości modelu generatywnej sztucznej inteligencji: inżynierię monitu i modyfikowanie parametrów wnioskowania. Stosując inżynierię monitu, m.in. formułując go na różne sposoby, umieszczając przykłady w kontekście lub zlecając rozumowanie krok po kroku, można sprawić, że model będzie generował trafniejsze, dokładniejsze uzupełnienia.

Dzięki inżynierii monitu, uczeniu kontekstowemu i parametrom wnioskowania można wiele osiągnąć, ale techniki te nie modyfikują wag modelu generatywnego. Dlatego czasami trzeba trenować model na własnych danych, aby lepiej „poznał” konkretną dziedzinę, i przygotować go do konkretnych zastosowań. Tym tematem zajmiemy się w następnych rozdziałach.

PROGRAM PARTNERSKI

— GRUPY HELION —

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

To fascynująca książka, rewelacyjna kompozycja niezwykle ważnych informacji, a także szczegółowych, praktycznych kodów, skryptów i instrukcji!

Jeff Barr, wiceprezes i główny popularyzator AWS

Podczas projektowania aplikacji opartych na generatywnej AI trzeba dokonywać wielu wyborów decydujących o jakości danych dostarczanych przez aplikację, jej opłacalności, skalowalności i niezawodności. Decyzje te są tym trudniejsze, że świat generatywnej AI zmienia się niezwykle szybko, a mity i błędne przeświadczenia dotyczące tej technologii mają się świetnie.

W tej niezwykle pragmatycznej książce, przeznaczonej dla dyrektorów technicznych, praktyków uczenia maszynowego, twórców aplikacji, analityków biznesowych, inżynierów i badaczy danych, znajdziesz skuteczne techniki używania sztucznej inteligencji. Zaznajomisz się z cyklem życia projektu opartego na generatywnej AI i jej zastosowaniami, a także metodami doboru i dostrajania modeli, generowania danych wspomagane wyszukiwaniem, uczenia przez wzmacnianie na podstawie informacji zwrotnych od człowieka, kwantyzacji, optymalizacji i wdrażania modeli. Poznasz szczegóły różnych typów modeli, między innymi dużych językowych (LLM), multimodalnych generujących obrazy (Stable Diffusion) i odpowiadających na pytania wizualne (Flamingo/IDEFICS).

Dowiedz się, jak:

- używać generatywnej AI w biznesie
- dobrać modele generatywnej AI
- stosować inżynierię monitu i uczenie kontekstowe
- dostrajać modele przy użyciu własnych zbiorów danych i techniki LoRA
- korzystać z agentów i akcji za pomocą bibliotek LangChain i ReAct
- tworzyć aplikacje na bazie usługi Amazon Bedrock

Chris Fregly jest głównym architektem rozwiązań w zakresie generatywnej AI w AWS i współautorem książki *Inżynieria danych na platformie AWS* (Helion, 2022).

Antje Barth jest główną programistką generatywnej AI w AWS i współautorką książki *Inżynieria danych na platformie AWS*.

Shelbee Eigenbrode jest główną architektką rozwiązań w zakresie generatywnej AI w AWS. Posiada ponad 35 patentów z różnych dziedzin techniki.

	KOD KORZYŚCI Sięgnij po więcej! ▶	
 helion.pl	ISBN 978-83-289-1474-2	
 HELION S.A. ul. Kościuszki 1c 44-100 Gliwice tel.: 52 230 98 63 helion@helion.pl		
9 788328 914742		
Cena: 79,00 zł		