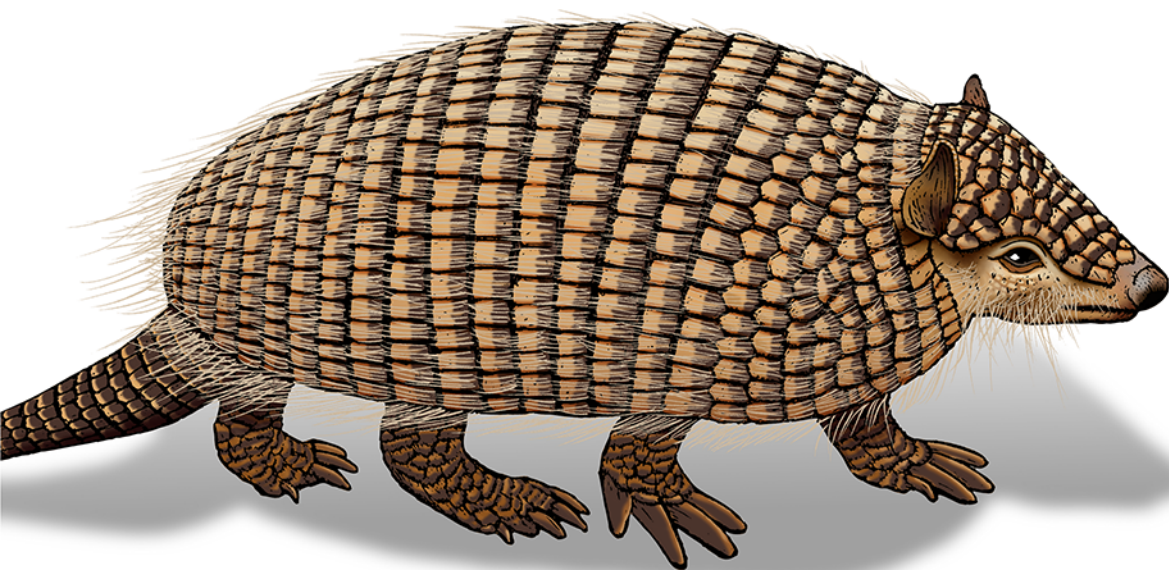


O'REILLY®

Helion 

Skuteczna inżynieria promptów

Przyszłościowe rozwiązania
dla rzetelnych wyników generatywnej AI



James Phoenix
Mike Taylor

Tytuł oryginału: Prompt Engineering for Generative AI: Future-Proof Inputs for Reliable AI Outputs

Tłumaczenie: Krzysztof Rychlicki-Kicior

ISBN: 978-83-289-1904-4

© 2025 Helion S.A.

Authorized Polish translation of the English edition of *Prompt Engineering for Generative AI*
ISBN 9781098153434 © 2024 Saxifrage, LLC and Just Understanding Data LTD.

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.

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.

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)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

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

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

Wprowadzenie	11
1. Pięć zasad promptowania	17
Omówienie pięciu zasad promptowania	20
1. Określ wytyczne	23
2. Określ format odpowiedzi	28
3. Przedstaw przykłady	30
4. Oceniaj jakość	34
5. Dziel pracę	45
Podsumowanie	50
2. Wprowadzenie do dużych modeli językowych do generowania tekstu	52
Czym są modele do generowania tekstu?	52
Reprezentacje wektorowe: język w liczbach	53
Architektura transformerów: orkiestracja związków kontekstowych	54
Probabilistyczne generowanie tekstu: mechanizm podejmowania decyzji	55
Krótka historia: wzrost znaczenia architektur transformerów	56
Wstępnie przeszkolony transformer generatywny OpenAI	58
GPT-3.5-turbo i ChatGPT	59
GPT-4	61
Google Gemini	61
Model Llama firmy Meta a otwarte oprogramowanie	62
Kwantyzacja i LoRA	63
Mistral	63
Anthropic: Claude	64
GPT-4V(ision)	64
Porównanie modeli	64
Podsumowanie	66

3. Standardowe zasady generowania tekstu z ChatGPT	67
Generowanie list	67
Generowanie list hierarchicznych	69
Kiedy unikać wyrażeń regularnych?	73
Generowanie danych w formacie JSON	73
Generowanie danych w formacie YAML	75
Filtrowanie dokumentów YAML	76
Obsługa nieprawidłowych dokumentów w formacie YAML	77
Generowanie różnorodnych formatów z użyciem ChatGPT	80
Spreparowane dane CSV	80
Wyjaśnij to jak pięciolatkowi	81
Uniwersalne tłumaczenia za pomocą LLM	82
Pytaj o kontekst	84
Wydzielenie stylu tekstu	87
Znalezienie pożądanych cech tekstu	87
Generowanie nowej treści za pomocą wyekstrahowanych cech	89
Ekstrakcja określonych cech tekstu za pomocą LLM	89
Podsumowywanie	89
Podsumowywanie a ograniczenia okna kontekstu	90
Podział tekstu na fragmenty	92
Zalety dzielenia tekstu na fragmenty	92
Scenariusze podziału tekstu na fragmenty	93
Niewłaściwe przykłady dzielenia tekstu na fragmenty	93
Strategie podziału	95
Wykrywanie zdań z użyciem spaCy	96
Tworzenie prostego algorytmu podziału w Pythonie	97
Podział za pomocą okna przesuwającego	98
Pakiety do dzielenia tekstu	100
Podział tekstu z biblioteką tiktoken	100
Kodowania	101
Zrozumienie procesu tokenizacji łańcuchów znaków	101
Szacowanie użycia tokenów w wywołaniach API czata	102
Analiza sentymentu	104
Sposoby usprawnienia analizy sentymentu	105
Ograniczenia i wyzwania analizy sentymentu	106
Od najmniejszych do największych	106
Planowanie architektury	107
Programowanie funkcji we Flasku	107
Dodawanie testów	108
Zalety techniki od najmniejszych do największych	108
Wyzwania, jakie wiążą się z techniką od najmniejszych do największych	109

Promptowanie z użyciem ról	109
Zalety promptowania z użyciem ról	110
Wyzwania, jakie wiążą się z promptowaniem z użyciem ról	111
Kiedy korzystać z promptowania z użyciem ról	111
Techniki promptowania GPT	112
Unikanie halucynacji dzięki tekstom źródłowym	112
Daj modelowi GPT „czas na przemyślenie”	113
Technika wewnętrznego monologu	114
Samooceńjące odpowiedzi modelu językowego	115
Klasyfikacja za pomocą dużych modeli językowych	116
Tworzenie modelu klasyfikacji	117
Głosowanie większością w klasyfikacji	118
Ewaluacja kryteriów	119
Metapromptowanie	122
Podsumowanie	125
4. Zaawansowane techniki generowania tekstu za pomocą LangChain	126
Wprowadzenie do LangChain	126
Konfiguracja środowiska	128
Modele czatowe	129
Strumieniowanie modeli czatowych	130
Generowanie wielu odpowiedzi z dużych modeli językowych	131
Szablony promptów LangChain	131
Język wyrażeń LangChain (LCEL)	132
Stosowanie PromptTemplate z modelami czatowymi	134
Parsery wyjścia	134
Ewaluacje LangChain	138
Wywołanie funkcji OpenAI	145
Współbieżne wywołanie funkcji	148
Wywoływanie funkcji w LangChain	149
Ekstrakcja danych za pomocą LangChain	151
Planowanie zapytań	151
Tworzenie szablonów promptów z kilkoma przykładami	153
Podejście z kilkoma przykładami o stałej długości	153
Formatowanie przykładów	154
Wybór promptów z kilkoma przykładami według długości	155
Ograniczenia promptów z kilkoma przykładami	157
Zapisywanie i wczytywanie promptów modeli językowych	157
Łączenie danych	158
Ładowarki dokumentów	160
Rozdzielacze tekstu	162

Podział tekstu według długości i liczby tokenów	163
Rekursywny podział według wielu znaków	164
Dekompozycja zadań	166
Łańcuchy promptów	168
Łańcuchy sekwencyjne	168
Ekstrakcja kluczy za pomocą funkcji itemgetter	170
Tworzenie struktury dla łańcuchów LCEL	175
Łańcuchy dokumentów	175
Łańcuch nadziania dokumentów	177
Łańcuchy oczyszczania	178
Mapowanie i redukcja	178
Łańcuch mapowania z rankingiem	179
Podsumowanie	180
5. Wektorowe bazy danych z FAISS i Pinecone	181
Retrieval Augmented Generation (RAG)	184
Wprowadzenie do osadzeń	185
Ładowanie dokumentów	193
Pozyskiwanie z pamięci za pomocą FAISS	195
RAG z użyciem frameworka LangChain	199
Wektorowe bazy danych w chmurze z użyciem Pinecone	201
Samoodpytywanie	208
Alternatywne mechanizmy pozyskiwania danych	212
Podsumowanie	214
6. Agenty autonomiczne z pamięcią i narzędziami	215
Łańcuch myśli	215
Agenty	217
Wnioskuj i działaj (ReAct)	219
Implementacja schematu wniosku i działaj	221
Stosowanie narzędzi	227
Duże modele językowe jako API (funkcje OpenAI)	228
Porównanie funkcji OpenAI i schematu ReAct	232
Przypadki użycia dla funkcji OpenAI	233
ReAct	233
Przypadki użycia dla schematu ReAct	234
Zestawy narzędzi dla agentów	234
Dostosowywanie standardowych agentów	236
Własne agenty w LCEL	237
Zasady użycia pamięci	239
Pamięć długoterminowa	239
Pamięć krótkoterminowa	240
Pamięć krótkoterminowa w agentach konwersacyjnych QA	240

Obsługa pamięci w LangChain	241
Zachowywanie stanu	242
Odpytywanie stanu	242
ConversationBufferMemory	242
Inne popularne rodzaje pamięci w LangChain	245
ConversationBufferWindowMemory	245
ConversationSummaryMemory	245
ConversationSummaryBufferMemory	246
ConversationTokenBufferMemory	246
Agent funkcji OpenAI z pamięcią	247
Zaawansowane frameworki agentowe	249
Agenty typu planuj i uruchamiaj	249
Drzewo myśli	250
Wywołania zwrotne	251
Globalne wywołania zwrotne (w konstruktorach)	253
Wywołania zwrotne żądań	253
Argument verbose	254
Które wywołanie zwrotne wybrać?	254
Zliczanie tokenów za pomocą LangChain	254
Podsumowanie	256

7. Wprowadzenie do modeli dyfuzyjnych przeznaczonych do generowania obrazów	257
OpenAI DALL-E	260
Midjourney	262
Stable Diffusion	265
Google Gemini	268
Generowanie filmów na podstawie tekstu	268
Porównanie modeli	268
Podsumowanie	269
8. Standardowe metody generowania obrazów z Midjourney	270
Modyfikatory formatu	270
Modyfikatory stylu w sztuce	274
Inżynieria odwrotna promptów	276
Dopalacze jakości	276
Prompty negatywne	278
Pojęcia ważne	281
Promptowanie z użyciem obrazków	284
Wmalowywanie	286
Domalowywanie	288

Spójne postaci	290
Przepisywanie promptów	292
Rozdzielenie memów	294
Mapowanie memów	298
Analiza promptów	300
Podsumowanie	302
9. Zaawansowane techniki generowania obrazów za pomocą Stable Diffusion	303
Uruchamianie modelu Stable Diffusion	303
Interfejs webowy AUTOMATIC1111	309
Img2Img	316
Skalowanie obrazków w górę	319
Tryb Interrogate CLIP	322
Wmalowanie i domalowanie w Stable Diffusion	322
ControlNet	325
Model segmentowania wszystkiego (SAM)	334
Dostrajanie DreamBooth	336
Dostrajacz modelu Stable Diffusion XL	343
Podsumowanie	346
10. Tworzenie aplikacji wspomaganych AI	348
Pisanie bloga przez AI	348
Badanie tematu	349
Wywiad ekspercki	352
Wygeneruj zarys	353
Generowanie tekstu	355
Styl pisania	357
Optymalizacja tytułu	360
Obrazki na blogu generowane przez AI	361
Interfejs użytkownika	366
Podsumowanie	368
Skorowidz	370

Wprowadzenie do dużych modeli językowych do generowania tekstu

W świecie sztucznej inteligencji niezwykle ważnym zagadnieniem w ostatnim czasie jest ewolucja dużych modeli językowych. W przeciwieństwie do mniej elastycznych poprzedników, duże modele językowe są w stanie radzić sobie ze znacznie większymi zbiorami danych i się z nich uczyć, co prowadzi do wykształcenia się zdolności do generowania tekstu niezwykle zbliżonego poziomem skomplikowania do możliwości człowieka. Modele radzą sobie z różnymi zadaniami, od pisania treści, przez automatyczne tworzenie oprogramowania, aż po tworzenie interaktywnych czatbotów odpowiadających na bieżąco użytkownikom.

Czym są modele do generowania tekstu?

Modele do generowania tekstu korzystają z zaawansowanych algorytmów w celu znalezienia znaczenia tekstu i generowania odpowiedzi często nieodróżnialnej od pracy człowieka. Jeśli miałeś doświadczenia z usługą ChatGPT (<https://chat.openai.com>) lub doświadczyłeś jej zdolności do generowania spójnych i przydatnych odpowiedzi, to poznałeś moc LLM w praktyce.

Zarówno w przetwarzaniu języka naturalnego, jak i w dużych modelach językowych podstawową jednostką językową jest token. **Tokeny** (<https://oreil.ly/3fOsM>) mogą reprezentować zdania, słowa, a nawet fragmenty słów, czyli zbiory znaków. Dobrą metodą wyrażania rozmiaru danych tekstowych jest określenie liczby tokenów danego tekstu. W języku angielskim 100 tokenów odpowiada mniej więcej 75 słowom. Warto zapamiętać tę proporcję, ponieważ każdy duży model językowy ma swoje ograniczenia wyrażone w tokenach — na podstawie tego przelicznika jesteśmy w stanie oszacować liczbę słów, którą możemy zmieścić w pojedynczym kontekście.

Tokenizacja to proces zamiany tekstu na tokeny — podstawowy etap przygotowania danych pod kątem zadań przetwarzania języka naturalnego (NLP). Tokenizację można wykonać na kilka sposobów, np. z wykorzystaniem metody *Byte-Pair Encoding* (kodowanie parami bajtów, BPE — <https://oreil.ly/iSOp7>), *WordPiece* lub *SentencePiece*. Każda z metod ma swoje unikatowe zalety i jest dostosowana do konkretnych przypadków użycia. BPE używa się głównie ze względu na wydajność obsługi w przypadku rozbudowanego słownika przy jednoczesnym zachowaniu rozsądnej liczby tokenów.

Metoda BPE rozpoczyna swoje działanie od potraktowania tekstu jako ciągu pojedynczych znaków. Z czasem najczęściej występujące ciągi znaków są grupowane w pojedyncze jednostki, zwane tokenami. Aby lepiej zrozumieć ten mechanizm, rozważmy słowo *jabłko*. Na początku BPE może traktować to słowo jako sześć niezależnych znaków — *j, a, b, ł, k, o*. Po analizie tekstów algorytm może zauważyć, że litera *b* często pojawia się po *a*, a zaś po *j*, z kolei *b* występuje przed *ł*, a *ł* przed *k*, co sprawia, że ciąg *jabłk* może zostać potraktowany jako pojedynczy token.

Takie podejście może pomóc dużym modelom językowym w generowaniu słów lub fraz, nawet jeśli nie występowały one często w danych treningowych — dzięki temu modele są w stanie dostosowywać się do różnych sytuacji i są bardziej uniwersalne.

Zrozumienie zasad działania dużych modeli językowych wymaga przyswojenia podstaw rządzących nimi reguł matematycznych. Choć obliczenia potrafią być skomplikowane, możemy uprościć kluczowe elementy, aby przedstawić intuicyjne zasady działania tych modeli. Zwłaszcza w kontekście aplikacji dla biznesu dokładność i stabilność dużych modeli językowych jest kluczowa.

Istotnym czynnikiem w osiągnięciu tej stabilności są fazy wstępnego uczenia (ang. *pretraining*) i dostrajania (ang. *fine-tuning*) w tworzeniu dużych modeli językowych. Na początku modele są uczone na ogromnych zbiorach danych w fazie wstępnego uczenia, co umożliwia im zrozumienie języka powszechnego, używanego w typowych, ogólnych sytuacjach. Z kolei w następującej potem fazie dostrajania modele dostosowuje się do konkretnych zadań, co pozwala zapewnić dokładne i wiarygodne wyniki dla aplikacji specjalistycznych.

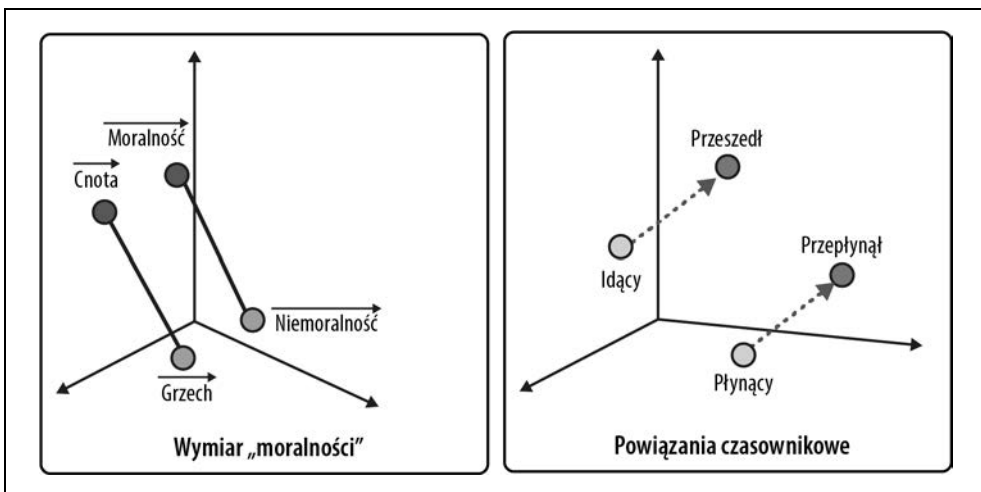
Reprezentacje wektorowe: język w liczbach

W świecie naturalnego przetwarzania języka słowa nie są jedynie zlepkami liter. Słowa mogą być poddane tokenizacji, a następnie przetwarzane w postaci liczbowej — wektorowej. Wektory to wielowymiarowe tablice liczb, które potrafią uchwycić związki semantyczne i składniowe:

$$w \rightarrow \mathbf{v} = [v_1, v_2, \dots, v_n]$$

Tworzenie wektorów słów, zwanych także **osadzeniami słów** (ang. *word embeddings*), polega na wykrywaniu zawiłych wzorców obecnych w języku. Podczas intensywnej fazy uczenia modele mają wykrywać te wzorce i się ich uczyć, by tym samym zapewnić, że słowa o podobnych znaczeniach będą znajdować się w przestrzeni wielowymiarowej blisko siebie (rysunek 2.1).

Piękno tego podejścia polega na zdolności do uchwycenia subtelnych związków pomiędzy słowami i obliczenia odległości między nimi. Jeśli przeanalizujemy osadzenia słów, to zauważymy bez problemu, że słowa o podobnych znaczeniach, takie jak *cnota* i *moralność* lub *idący* i *przeszedł*, są zlokalizowane blisko siebie. Bliskość w przestrzeni osadzeń staje się niezwykle istotna w rozmaitych zadaniach z zakresu NLP, zapewniając modelom zrozumienie kontekstu, semantyki i skomplikowanej sieci powiązań, które tworzą język.



Rysunek 2.1. Semantyczna bliskość wektorów słów wewnątrz przestrzeni osadzeń słów

Architektura transformerów: orkiestracja związków kontekstowych

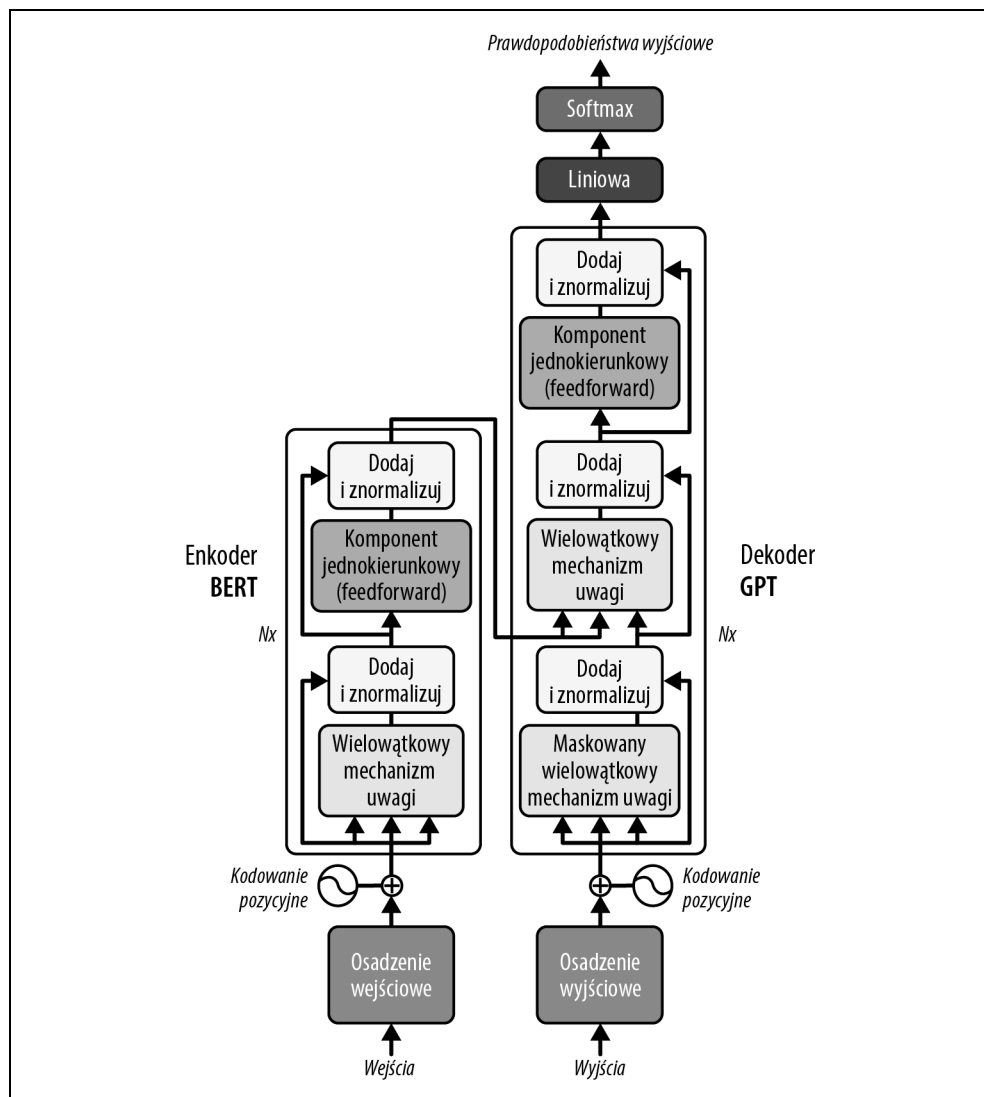
Zanim bardziej zagłębimy się w szczegóły działania architektury transformerów, musimy zrozumieć podstawy. Jeżeli mamy dowolne zdanie, np. *kot usiadł na podłodze*, to każde słowo w zdaniu zostanie przetworzone na reprezentację wektorową. Słowo *kot* zostaje przekonwertowane na serię liczbę, podobnie jak *usiadł*, *na* i *podłodze*.

Jak przekonasz się w dalszej części rozdziału, architektura transformerów przyjmuje te wektory słów i znajduje związki występujące pomiędzy nimi — zarówno pod kątem struktury (składni), jak i znaczenia (semantyki). Istnieje wiele rodzajów transformerów — rysunek 2.2 przedstawia architektury BERT i GPT. Co więcej, transformer nie rozpatruje słów niezależnie — w naszym przykładzie wie, że słowo *kot* jest w tym zdaniu w pewien sposób powiązane z *usiadł* i *podłodze*.

Transformer w trakcie przetwarzania wektorów korzysta z matematycznych operacji do rozumienia związków między słowami i w ten sposób tworzy nowe wektory o bogatej informacji kontekstowej:

$$\mathbf{v}'_i = \text{Transformer}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m)$$

Jedną z ważniejszych możliwości transformerów jest zdolność do rozumienia subtelnych znaczeń słów, zależnych od kontekstu. Mechanizm samouwagi (<https://oreil.ly/xuovP>) pozwala każdemu słowu zapoznać się z pozostałymi słowami w pobliżu, aby lepiej zrozumieć kontekst. Można powiedzieć, że każde słowo głosuje na znaczenie innych słów, aby ustalić swoje znaczenie. Analizując całe zdanie, transformery są w stanie dokładnie określić rolę i znaczenie każdego słowa, dzięki czemu *ich zdolności interpretacyjne są bardziej dopasowane do kontekstu*.



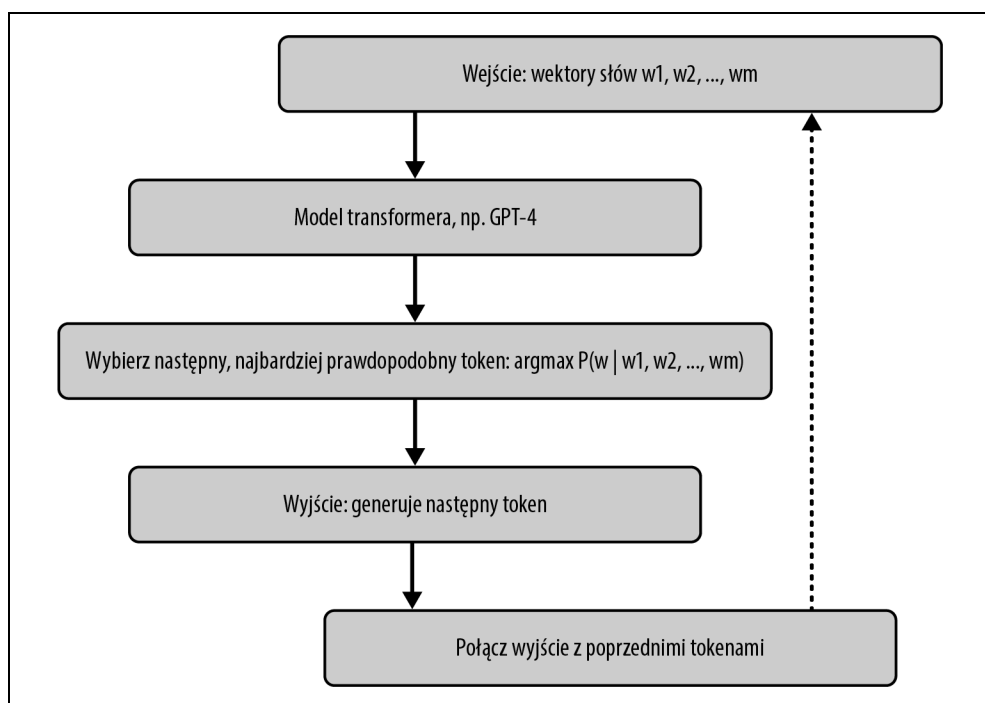
Rysunek 2.2. BERT stosuje enkoder do danych wejściowych, a GPT — dekodery do danych wyjściowych

Probabilistyczne generowanie tekstu: mechanizm podejmowania decyzji

Gdy transformer rozumie kontekst danego tekstu, może przejść do generowania nowego, korzystając z pojęcia prawdopodobieństwa (ang. *likelihood/probability*). W sensie matematycznym: model oblicza prawdopodobieństwo wystąpienia kolejnego słowa, biorąc pod uwagę bieżącą sekwencję słów, i wybiera to najbardziej prawdopodobne:

$$w_{\text{nast}} = \operatorname{argmax} P(w \mid w_1, w_2, \dots, w_m)$$

Powtarzając ten proces (rysunek 2.3), model jest w stanie wygenerować spójny i kontekstowo istotny tekst.



Rysunek 2.3. Ogólna metoda generowania tekstu przez modele transformatorów, takich jak GPT-4

Mechanizmy stojące za sukcesem dużych modeli językowych pochodzą z matematyki wektorów, transformacji liniowej i modeli probabilistycznych. Pod względem implementacji operacje te są niezwykle intensywne obliczeniowo, niemniej warto pamiętać, że główne pojęcia związane z funkcjonowaniem transformatorów opierają się na wspomnianych matematycznych zasadach, co z pewnością ułatwi zrozumienie technicznych aspektów działania transformatorów przez interesariuszy, np. z biznesu.

Krótko o historii: wzrost znaczenia architektur transformatorów

Modele językowe takie jak ChatGPT (GPT — *Generative Pretrained Transformer*, **wstępnie przeszkolony transformer generatywny**) nie pojawiły się znikąd. Ich obecność to efekt wielu lat postępu w dziedzinie przetwarzania języka naturalnego (NLP), zwłaszcza w drugiej dekadzie tego stulecia. Kluczowym elementem tego postępu było wprowadzenie architektur transformatorów, opisanych szczegółowo w przełomowym artykule *Attention Is All You Need* („Uwaga to wszystko, czego potrzebujesz”, <https://oreil.ly/6NNbg>) zespołu Google Brain.

Prawdziwym przełomem w architekturach transformerów było wprowadzenie mechanizmu uwagi (ang. *attention*). Tradycyjne modele przetwarzały tekst sekwencyjnie, co ograniczało możliwość zrozumienia struktury języka, zwłaszcza na przestrzeni długich odstępów w ramach tekstu. Mechanizm uwagi zmienił tę sytuację: umożliwił modelom powiązanie ze sobą odległych słów, niezależnie od ich położenia w tekście. Takie podejście było naprawdę przełomowe — dzięki niemu słowa i ich kontekst nie musiały być przemieszczane w obrębie całego modelu, aby mieć na siebie wpływ. Takie podejście nie tylko znacząco poprawiło zdolność rozumienia tekstu przez modele, ale także zwiększyło ich wydajność.

Mechanizm uwagi odgrywa najważniejszą rolę w umożliwieniu modelom wykrywania zależności pomiędzy słowami znajdującymi się w dużej odległości od siebie. Ta zmiana stała się kluczowa w generowaniu odpowiedzi, które są nie tylko kontekstowo dokładne, ale także spójne na dłuższym dystansie.

Według pioniera i edukatora AI Andrew Ng (<https://oreil.ly/JQd53>) duża liczba wczesnych badań prowadzonych w zakresie NLP była istotnie dofinansowana przez agencje wywiadu wojskowego Stanów Zjednoczonych. Duże zainteresowanie tych agencji w zakresie narzędzi do tłumaczenia maszynowego czy rozpoznawania mowy, głównie pod kątem działań wywiadowczych, siłą rzeczy przyczyniło się do rozwoju możliwości wykraczających poza tłumaczenie.

Uczenie dużych modeli językowych wymaga sporych zasobów obliczeniowych. Modele są „karmione” ogromnymi ilościami danych — rzędu terabajtów, a nawet petabajtów — w których skład wchodzi treści z Internetu, artykuły naukowe, książki i bardziej specjalistyczne zbiory danych. Należy przy tym pamiętać, że dane używane do trenowania dużych modeli językowych mogą charakteryzować się różnymi *tendencjami* (ang. *bias*), *wynikającymi wprost ze źródeł*. W związku z tym użytkownicy powinni zachować szczególną ostrożność i ciągły nadzór nad wdrażaniem modeli, by zapewnić możliwość tworzenia bezpiecznych i etycznych aplikacji AI.

Model GPT-4, należący do OpenAI, dysponuje podobno mniej więcej 1,7 biliona parametrów (<https://oreil.ly/pZvMo>), co stanowi odpowiednik arkusza Excela o wielkości 30 tysięcy boisk do piłki nożnej. **Parametry** — w kontekście sieci neuronowych — to wagi i tendencje dostosowywane w procesie uczenia, które pozwalają modelowi na rozpoznawanie i generowanie złożonych wzorców na podstawie użytych danych treningowych. Koszt nauczania modelu GPT-4 jest szacowany na 63 miliony dolarów (https://oreil.ly/_NAq5) i same dane treningowe byłyby w stanie wypełnić 650 kilometrów regałów wypełnionych książkami (<https://oreil.ly/D7jL5>).

Osiągnięcie takich efektów wymagało sporych inwestycji ze strony firm technologicznych, takich jak Microsoft, Meta czy Google, co uczyniło rozwój dużych modeli językowych niezwykle drogim przedsięwzięciem.

Wzrost znaczenia LLM-ów zwiększył zapotrzebowanie także od strony sprzętu, zwłaszcza na produkty firm specjalizujących się w kartach graficznych z wyspecjalizowanymi układami (GPU). Kluczowym graczem stała się Nvidia, utożsamiana wręcz z wysoko wydajnymi kartami graficznymi, niezbędnymi do uczenia dużych modeli językowych.

Zapotrzebowanie na potężne, wydajne procesory graficzne niezwykle wzrosło wraz z budową coraz większych i coraz bardziej złożonych modeli. Nie chodzi przy tym tylko o samą moc

obliczeniową. GPU muszą być także dopasowane do zadań kluczowych z punktu widzenia uczenia maszynowego, takich jak operacje na tensorach. **Tensory** — ponownie w kontekście uczenia maszynowego — to wielowymiarowe tablice danych, a operacje na nich stanowią fundament wszelkich obliczeń związanych z sieciami neuronowymi. Skoncentrowanie się na tych wyspecjalizowanych działaniach przyczyniło się do pojawienia się specjalistycznego sprzętu, takiego jak karty Nvidia H100 Tensor Core, w pełni przystosowane do operacji związanych z uczeniem maszynowym.

Przytłaczający popyt często przewyższa podaż, co naturalnie zwiększa ceny sprzętu. W związku z tym rynek GPU stał się niezwykle konkurencyjnym, ale i zyskownym obszarem, w którym zróżnicowana klientela — od gigantów technologicznych do pracowników naukowych — stara się opracować najbardziej zaawansowany sprzęt.

Ogromny popyt przyczynił się do powstania produktów wykraczających poza GPU. Firmy koncentrują się na tworzeniu sprzętu przeznaczonego do zadań związanych ze sztuczną inteligencją, takiego jak TPU (ang. *Tensor Processing Unit* — jednostka przetwarzania tensorów) firmy Google, a wszystko to w celu zaspokojenia rosnącego zapotrzebowania modeli AI na moc obliczeniową.

Ten ciągle zmieniający się krajobraz nie tylko podkreśla silne związki między oprogramowaniem a sprzętem, ale także pokazuje efekt swoistej gorączki złota związanej z dużymi modelami językowymi. Ogromne inwestycje i innowacje trafiają do różnych branż, ale zwłaszcza tych, które oferują komponenty niezbędne do dalszego rozwoju tych modeli.

Wstępnie przeszkolony transformer generatywny OpenAI

Organizacja OpenAI (<https://openai.com>), którą założono z myślą o przyniesieniu całej ludzkości korzyści z wdrażania sztucznej inteligencji, stała się jednym z pionierów rewolucji AI. Jednym z największych jej dokonań jest seria modeli GPT, która znacząco zmieniła nasze postrzeganie tego, co mogą nam dać duże modele językowe.

Oryginalny model GPT autorstwa OpenAI był czymś więcej niż tylko efektem badań. Był to intrygujący przykład możliwości architektur opartych na transformerach. Model ten stanowił pierwszy krok w kierunku umożliwienia maszynom zrozumienia i generowania języka tak, jak robią to ludzie, i położył fundament pod przyszły rozwój technologii.

Prezentacji modelu GPT-2 towarzyszyły zarówno oczekiwania, jak i swego rodzaju niepokój. Mając świadomość dużych możliwości modelu, OpenAI początkowo wahała się, czy go udostępnić, w obawie przed nadużyciami. Obawy natury etycznej stanowiły centrum sporu wokół modelu GPT-2, co może wydawać się osobliwe w porównaniu z możliwościami obecnych modeli. Opublikowanie projektu w formie otwartego oprogramowania (<https://openai.com/research/transformer-lstm>) nie oznaczało jedynie opublikowania kodu. Ta decyzja pozwoliła firmom i badaczom skorzystać ze wstępnie przeszkolonych modeli jako komponentów przy budowie własnych rozwiązań AI, bez konieczności budowania takich rozwiązań od podstaw. Dzięki temu dostęp do wysokiej jakości mechanizmów przetwarzania języka naturalnego stał się możliwy dla wszystkich, co umożliwiło innowacje w różnych obszarach.

Po opublikowaniu modelu GPT-2 OpenAI skupiła się na przedstawieniu płatnych, zamkniętych modeli. GPT-3 uznaje się za ogromny postęp w dziedzinie dużych modeli językowych. Model ten przyciągnął uwagę mediów, nie tylko ze względu na stopień zaawansowania technicznego, ale także na implikacje, jakie niósł on ze sobą dla całego społeczeństwa. Model ten potrafił generować tekst w tak przekonujący sposób, że trudno było odróżnić go od wytworów człowieka. Od pisania rozbudowanych tekstów literackich, aż po generowanie działających fragmentów kodu — GPT-3 pokazał właściwie nieograniczony potencjał AI.

GPT-3.5-turbo i ChatGPT

Napędzona znaczącą inwestycją dokonaną przez Microsoft, OpenAI przedstawiła model GPT-3.5-turbo, zoptymalizowaną wersję doskonałego poprzednika. Dzięki dotacji rzędu miliarda dolarów (<https://oreil.ly/1C8qm>) otrzymanej od Microsoftu w 2019 roku, po której nastąpiła inwestycja o wielkości 13 miliardów dolarów w zamian za pakiet 49% udziałów w komercyjnej spółce powiązanej z OpenAI, możliwe było opracowanie modelu GPT-3.5-turbo, który osiągnął jeszcze większą skuteczność, ale i umożliwił tańsze działanie, dzięki czemu dużych modeli językowych można było używać w znacznie większej liczbie przypadków.

Organizacja chciała uzyskać jak najwięcej informacji zwrotnej od szerokiego grona użytkowników, by dostroić swój model, dlatego uruchomiła usługę ChatGPT (<https://chat.openai.com>). W przeciwieństwie do siostrzanych modeli ogólnego przeznaczenia, ChatGPT dostrojono (<https://oreil.ly/6ib-Q>) pod względem umiejętności prowadzenia konwersacji, co umożliwiło prowadzenie dialogu między ludźmi a maszynami w sposób naturalny i owocny.

Na rysunku 2.4 jest pokazany proces uczenia usługi ChatGPT, obejmujący trzy główne kroki:

Zebranie danych demonstracyjnych

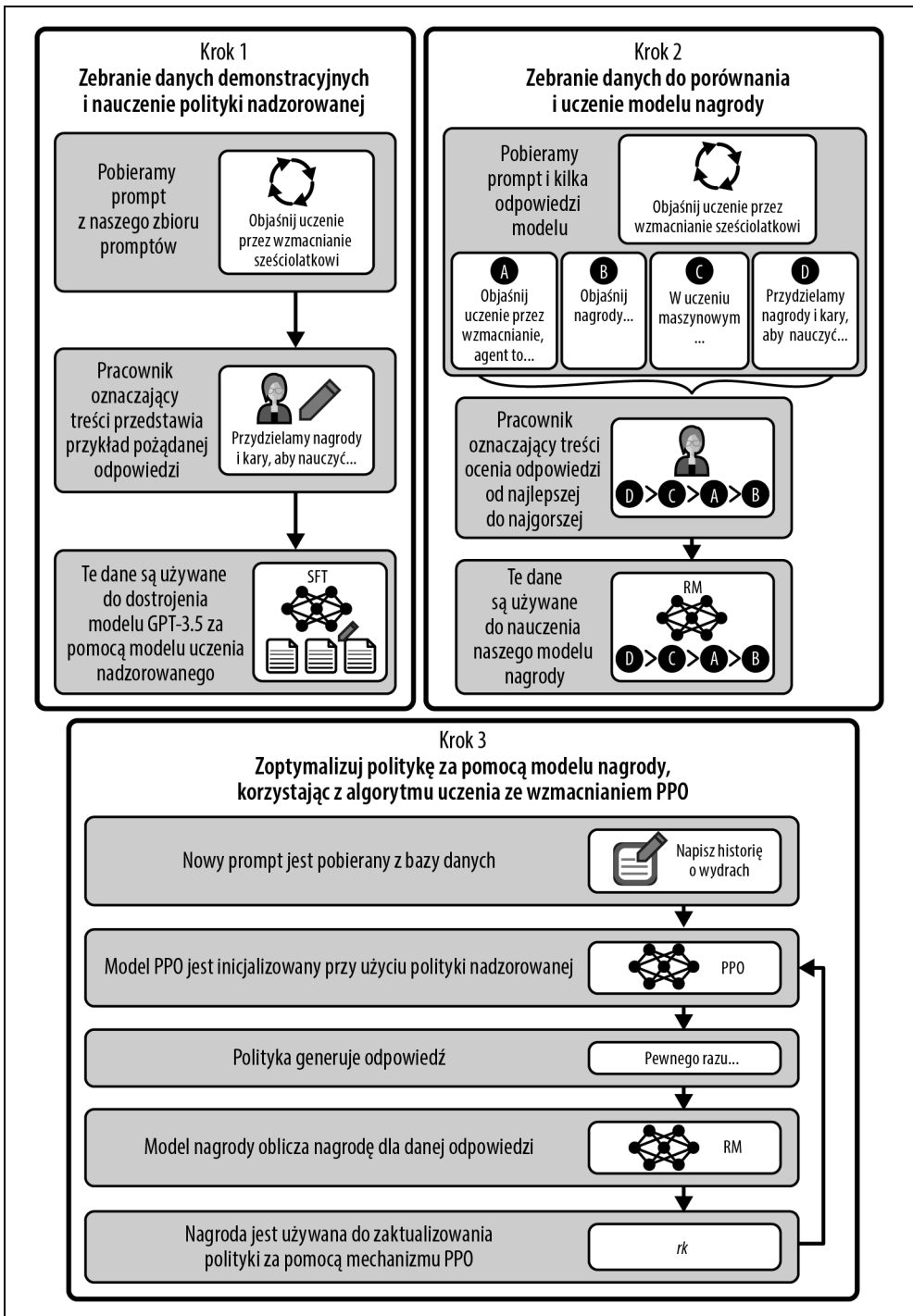
W tym kroku pracownicy przedstawiają przykłady pożądanego zachowania modelu na pewnej selekcji promptów. Pracownicy ci otrzymują konkretne wytyczne, aby mogli oni oznaczać prompty dokładnie.

Uczenie z nadzorem (uczenie nadzorowanej polityki)

Przykładowe dane zebrane w poprzednim kroku są używane do dostrajania wstępnie przeszkolonego modelu GPT-3 za pomocą mechanizmu uczenia z nadzorem (ang. *supervised learning*). W takim trybie modele są uczone z użyciem oznaczonego zbioru danych, z uwzględnieniem poprawnych odpowiedzi dla zadanych problemów. Ten krok pomaga modelowi nauczyć się podążać zgodnie z danym zestawem instrukcji i generować wyjście zgodne z oczekiwaniami.

Zebranie danych porównawczych i uczenie przez wzmacnianie

W tym kroku zbiera się zestaw odpowiedzi modelu, który następnie oceniają pracownicy oznaczający treści. W ten sposób jest uczony model nagrody, który pozwala przewidywać, jak pracownicy oznaczający treści oceniliby dane odpowiedzi. Na zakończenie wprowadza się mechanizm uczenia przez wzmacnianie (ang. *reinforcement learning*), a konkretnie: przybliżoną optymalizację polityki (ang. *Proximal Policy Optimization* — PPO). PPO pozwala optymalizować politykę nadzorowania w celu maksymalizacji nagrody uzyskanej z modelu nagrody.



Rysunek 2.4. Proces dostrojenia modelu ChatGPT

Proces uczenia pozwala modelowi ChatGPT uzyskać zachowanie zbliżone do ludzkiego. Zastosowanie uczenia przez wzmacnianie wraz z uwzględnieniem czynnika ludzkiego w całym procesie pozwoliło stworzyć model pomocny, szczery i bezpieczny — w porównaniu ze wstępnie uczonym modelem GPT-3.

Według badań UBS (<https://oreil.ly/2Ivq2>) do stycznia 2023 r. ChatGPT ustanowił rekord 100 milionów aktywnych użytkowników i stał się zarazem najszybciej rozwijającą się aplikacją konsumencką w historii Internetu. ChatGPT stał się domyślnym wyborem w implementacji mechanizmów obsługi użytkownika, wirtualnych asystentów i wielu innych aplikacji, które wymagają prowadzenia konwersacji na poziomie człowieka.

GPT-4

W roku 2024 OpenAI opublikowała model GPT-4, który świetnie sprawdza się w rozumieniu złożonych zapytań oraz generowaniu istotnych i spójnych informacji tekstowych. Dla przykładu, GPT-4 uzyskał na egzaminie prawniczym wynik w granicach 90. percentyla, 298 na 400 punktów. Obecnie model GPT-3.5-turbo jest darmowy do użycia w ramach usługi ChatGPT, natomiast GPT-4 wymaga uiszczenia miesięcznej opłaty (<https://oreil.ly/UOEBM>).

GPT-4 korzysta z podejścia tzw. mieszanki ekspertów (ang. *mixture of experts*, <https://oreil.ly/v45LZ>), w którym model uwzględnia więcej niż jeden wynik wnioskowania (ang. *inference*), aby wygenerować jeszcze bardziej dokładne i istotne wyniki.

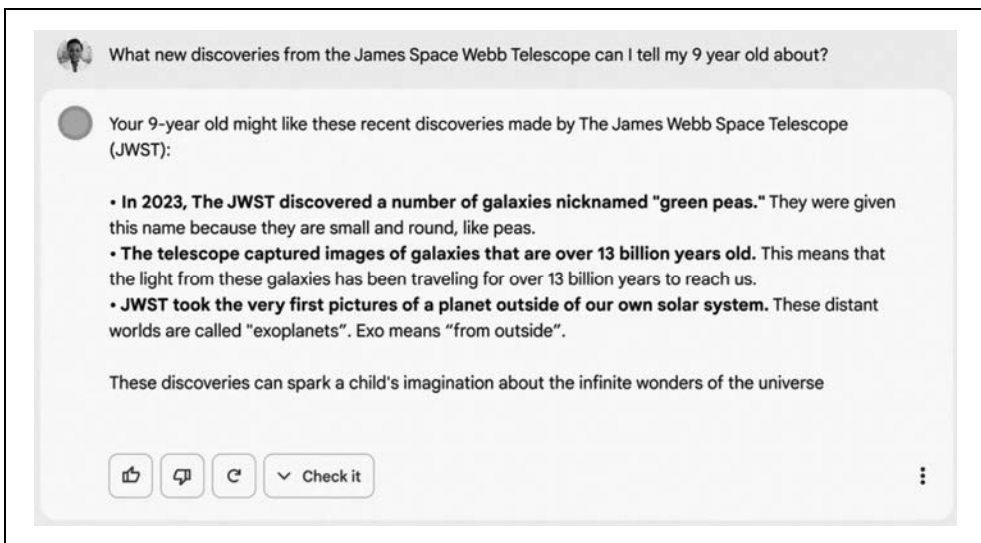
13 maja 2024 r. OpenAI opublikowała GPT-4o (<https://oreil.ly/4ttmq>), zaawansowany model zdolny do przetwarzania danych wejściowych tekstowych, dźwiękowych i graficznych w czasie rzeczywistym. Zapewnia on zwiększoną efektywność, zwłaszcza w kontekście danych audio i wideo. Jest także szybszy i tańszy niż poprzednicy, głównie dzięki możliwości przetwarzania trzech różnych form danych wejściowych w jednej sieci neuronowej.

Google Gemini

Google utraciło część ruchu (i zarazem rynku) wyszukiwarek internetowych właśnie w związku z pojawieniem się modelu ChatGPT. W odpowiedzi, 21 marca 2023 r., opublikowało własny model Bard. Na początku zdarzały się problemy (<https://oreil.ly/Sj24h>), a średnia jakość odpowiedzi zdecydowanie odbiegała od usług udostępnianych przez ChatGPT (rysunek 2.5).

Z czasem dodano kilka nowych funkcji, takich jak generowanie kodu, AI wizualne, wyszukiwanie w czasie rzeczywistym, a także głosowe — co przybliżyło go pod względem jakości do modelu ChatGPT.

14 marca 2023 r. Google opublikowało PaLM API (<https://oreil.ly/EbI8->), dając dostęp programistyczny z poziomu usługi Google Cloud Platform. W kwietniu 2023 r. AWS (Amazon Web Services) opublikowało podobne usługi, takie jak Amazon Bedrock (<https://oreil.ly/4fNQX>)



Rysunek 2.5. Halucynacje modelu Bard na temat teleskopu Jamesa Webba

czy Amazon's Titan FMs (<https://oreil.ly/FJ-7D>). Google w wersji 1.5 (w lutym 2024 r.) zmieniło nazwę swojego rozwiązania z Bard na Gemini (<https://oreil.ly/EO42O>), a równocześnie jego użytkownicy zaczęli uzyskiwać wyniki zbliżone do tych, które zapewniał GPT-4.

Google opublikowało także dwa mniejsze otwarte modele (<https://oreil.ly/LWIww>) oparte na tej samej architekturze, co Gemini. Rozwiązanie OpenAI nie stanowi już jedynego, oczywistego wyboru w zakresie dużych modeli językowych dla inżynierów oprogramowania tworzących aplikacje z ich użyciem.

Model Llama firmy Meta a otwarte oprogramowanie

Podejście firmy Meta do modeli językowych różni się znacząco od podejścia konkurencji. Kolejne wersje swojego kluczowego modelu Llama — pierwszą (<https://oreil.ly/LroPn>), drugą (<https://oreil.ly/NeZLw>) i trzecią (<https://oreil.ly/Vwlo->), opublikowała na zasadach otwartego oprogramowania. Wynika to z faktu, że Meta chce uzyskać bardziej inkluzywny i zachęcający do współpracy ekosystem tworzenia rozwiązań AI.

Otwarty charakter modeli Llama 2 i 3 ma istotne konsekwencje dla szeroko rozumianego przemysłu IT, zwłaszcza dla korporacji. Przejrzystość i etos współpracy napędzają innowacje, ponieważ wszelkie problemy i podatności mogą być szybko zidentyfikowane i rozwiązane dzięki globalnej społeczności programistów. W miarę jak modele stają się bardziej rozwinięte i bezpieczne, korporacje są w stanie ufać w zasadność ich wdrażania.

Strategia otwartego oprogramowania firmy Meta nie tylko demokratyzuje dostęp do najistotniejszych technologii AI, ale także może mieć istotny wpływ na cały przemysł. Rozpropagowanie stosowanego w modelach Llama transparentnego, zdecentralizowanego procesu opartego na

szerokiej współpracy może wpłynąć na przyszłość generatywnej AI. Wspomniane modele są dostępne w wersjach o 7, 8 i 70 miliardach parametrów w usługach AWS, Google Cloud, Hugging Face i wielu innych.

Otwarty charakter tych modeli można traktować jako miecz obusieczny. Z jednej strony ułatwia dostęp do technologii, zwłaszcza mniejszym firmom i organizacjom. Nawet indywidualni programiści są w stanie dołożyć swoją cegiełkę do rozwoju modeli, a także mogą stosować je w prawdziwych projektach, np. aplikacjach biznesowych. Ten rodzaj zdecentralizowanej innowacji może prowadzić do przełomów, które nie wystąpiłyby w odseparowanym środowisku pojedynczej organizacji, a tym samym zwiększyć przyszłe możliwości tych modeli i rozszerzyć ich zastosowania.

Z drugiej strony, ta sama otwartość może powodować pewne ryzyko, np. przestępcy mogą korzystać z tych technologii do niecnych celów. Jest to jedna z obaw, które wyrażają organizacje takie jak OpenAI, sugerując, że pewien poziom kontroli dostępu i ograniczeń może w istocie pomóc w ograniczeniu możliwości tworzenia niebezpiecznych aplikacji.

Kwantyzacja i LoRA

Jednym z kluczowych aspektów modeli otwartych jest stosowanie pojęcia kwantyzacji (<https://oreil.ly/bkWXk>) i LoRA (*low-rank approximations*, aproksymacje niskiego rzędu — <https://oreil.ly/zORsB>). Obie techniki pozwalają na dopasowanie modeli do urządzeń o mniejszych możliwościach sprzętowych. Kwantyzacja pomaga zmniejszyć dokładność obliczeniową parametrów modelu, co wpływa na całkowity rozmiar modelu bez istotnej straty w jakości działania. LoRA z kolei pozwala optymalizować architekturę sieci, by uczynić ją bardziej wydajną w działaniu na sprzęcie konsumenckim.

Tego rodzaju optymalizacje sprawiają, że dostrajanie dużych modeli językowych staje się coraz bardziej dostępne na sprzęcie konsumenckim. Jest to niezwykle ważne, bo daje możliwość szerokiego prowadzenia eksperymentów na modelach i dostosowywania ich. Nie trzeba mieć już własnego centrum danych — zwykli programiści, małe firmy czy startupy mogą pracować z tymi modelami nawet w mocno ograniczonych środowiskach.

Mistral

Mistral 7B, wytwór francuskiego startupu Mistral AI (<https://mistral.ai>), jawi się jako prawdziwy kombajn w dziedzinie generatywnej sztucznej inteligencji — m.in. dzięki 7,3 miliarda parametrów. Nie tylko rozmiar ma tu znaczenie — jego siła tkwi w efektywności i zdolnościach, co pozwala mieć nadzieję na świetlaną przyszłość otwartych dużych modeli językowych i ich zastosowań. Kluczowa dla efektywności jest implementacja mechanizmu przesuwającego okna uwagi, opublikowana na otwartej licencji Apache. Wielu inżynierów AI dostraja swoje rozwiązania na podstawie tego modelu, wliczając w to imponujący model Zephyr 7b beta (https://oreil.ly/Lg6_r). Warto rozważyć także model Mixtral 8x7b (<https://oreil.ly/itsJG>), czyli model mieszkanki ekspertów (podobny do architektury GPT-4), który osiąga rezultaty zbliżone do GPT-3.5-turbo.

Więcej szczegółów i aktualne porównanie modeli otwartych, a także metryki wydajności znajdziesz na stronie Chatbot Arena Leaderboard (<https://oreil.ly/ttiji>), udostępnianej przez Hugging Face.

Anthropic: Claude

Model Claude 2 (<https://claude.ai/login>), opublikowany 11 lipca 2023 r., odróżnia się od innych dużych modeli językowych, takich jak ChatGPT czy LLaMA, za sprawą tzw. konstytucyjnego AI (ang. *constitutional AI*) związanego z bezpieczeństwem i nastawieniem (<https://oreil.ly/Tim9W>) — uczenie modelu odbywa się za pomocą listy reguł lub wartości. Istotnymi usprawnieniami w modelu Claude 2 jest rozszerzone okno kontekstu wielkości 100 000 tokenów, a także możliwość wysyłania plików. W świecie generatywnej sztucznej inteligencji okno kontekstu oznacza ilość tekstu (lub danych), którą model jest w stanie mieć na uwadze w trakcie generowania odpowiedzi. Im większe okno kontekstu, tym szerszy jest kontekst, na podstawie którego model wnioskuje i generuje odpowiedzi.

Te innowacje społeczność inżynierów AI przyjęła bardzo entuzjastycznie, pojawiły się bowiem dzięki temu nowe i bardziej wyszukane scenariusze użycia tego modelu. Na przykład możliwość przetwarzania większej ilości informacji pozwala na podsumowywanie dłuższych dokumentów lub prowadzenie bardziej rozbudowanych konwersacji. Ta przewaga nie trwała zbyt długo, ponieważ zaledwie pół roku później OpenAI opublikowała wersję modelu GPT-4 o wielkości okna kontekstu 128 000 tokenów (<https://oreil.ly/BWxrn>). Nie zmienia to faktu, że rywalizacja pozytywnie wpłynęła na rozwój całego sektora.

Kolejna generacja modelu Claude zawiera Opus (<https://oreil.ly/NH0jh>) — pierwszy model, który jest zdolny rywalizować z GPT-4 pod względem inteligencji, a także Haiku — mniejszy model, który jest niezwykle szybki, a także dość tani, ponieważ oferuje 1 000 000 tokenów w cenie 25 centów (w momencie gdy pisaliśmy te słowa, była to połowa kosztu użycia usługi GPT-3.5-turbo).

GPT-4V(ision)

Istotnym krokiem naprzód, poczynionym 23 września 2023 r., było rozszerzenie przez OpenAI modelu GPT-4 o możliwość przetwarzania obrazów za pomocą mechanizmu Vision. Innowację tę uwzględniono w interfejsie ChatGPT, który obecnie obsługuje jako wejście zarówno obrazy, jak i teksty. Ten krok pokazuje pewien szerszy trend, jakim są **modele multimodalne**, które mogą bez problemu przetwarzać i rozumieć różne rodzaje danych, np. obrazy i tekst, w ramach jednego kontekstu.

Porównanie modeli

Rynek modeli LLM w momencie, gdy piszemy te słowa, jest zdominowany przez OpenAI, a model GPT-4 powszechnie uznawano za lidera rynku. Najbliższym konkurentem jest Anthropic, a sporo nadziei budzą też mniejsze, ale otwarte modele, takie jak Llama i Mistral — zwłaszcza pod kątem

dostrajania. Chociaż zdaniem ekspertów OpenAI będzie dalej dostarczać najlepsze modele w przyszłości, to modele otwarte mogą okazać się *wystarczająco dobre* do coraz większej grupy zadań. To sprawi, że zadania związane z AI mogą być coraz częściej realizowane przez modele dostrajane lokalnie. Biorąc pod uwagę postępy w wydajności modeli i kwantyzację (metody używane w celu uzyskania kompromisu między dokładnością działania a rozmiarem i kosztem obliczeń), możemy kiedyś dojść do momentu, w którym duże modele językowe będzie można uruchamiać nawet na urządzeniach mobilnych.

Na razie najlepszą metodą porównania modeli jest uruchomienie tego samego prompta w wielu modelach i porównanie odpowiedzi. Jedną niezmienną regułą jest to, że GPT-4 radzi sobie znacznie lepiej w wykonywaniu instrukcji, co widać w poniższym przykładzie — jest to jedyny model, który wygenerował odpowiedź w prawidłowym formacie, z nazwami dopasowanymi do przykładów (zaczynających się od litery *i*).

Wejście:

Czy możesz wymyślić listę nazw dla butów, które mogą pasować do dowolnego rozmiaru stopy w języku polskim, tak jakby zaproponował je Steve Jobs?

Zwróć wyniki w postaci listy przedzielonej przecinkami, w następującym formacie:

Opis produktu: But, który pasuje do dowolnego rozmiaru stopy

Nazwy produktów: [lista trzech nazw produktów]

Przykłady

Opis produktu: Lodówka, która wydaje piwo

Nazwy produktów: iBarFridge, iFridgeBeer, iDrinkBeerFridge

Opis produktu: Zegarek, który podaje dokładny czas w przestrzeni kosmicznej

Nazwy produktów: iNaut, iSpace, iTime

Opis produktu: Automat do przygotowania mlecznych koktajli w domu

Nazwy produktów: iShake, iSmoothie, iShake Mini

Wyjście (GPT-4):

Opis produktu: But, który pasuje do dowolnego rozmiaru stopy

Nazwy produktów: iStep, iFitBoot, iComfortShoe

Wyjście (Claude 3):

Oto lista nazw dla butów, które pasują do dowolnego rozmiaru stopy, w stylu Steve'a Jobsa:

Opis produktu: But, który pasuje do dowolnego rozmiaru stopy

Nazwy produktów: iMorph, iAdapt, iFlex

Wyjście (Llama 3 70b):

Oto lista nazw dla butów, które mogą pasować do dowolnego rozmiaru stopy w języku polskim, w stylu Steve Jobsa:

Opis produktu: But, który pasuje do dowolnego rozmiaru stopy

Nazwy produktów: iFit, UniStep, AdaptWalk

Podsumowanie

Podróż dużych modeli językowych od ich powstania aż po obecną, zaawansowaną postać to opowieść o ciągłej innowacji, współpracy i intensywnej konkurencji. Wraz z ewolucją obecnych modeli zwiększa się prawdopodobieństwo, że staną się one integralną częścią naszego codziennego życia, zmieniając sposób, w jaki korzystamy z technologii, a nawet w jaki komunikujemy się ze sobą.

Zrozumienie kontekstu historycznego, a także możliwości tych modeli pozwoli Ci lepiej docenić narzędzia, jakimi dysponujemy — od inżynierii promptów aż po rozwój inteligentnych agentów wirtualnych. Warto pamiętać, że choć modele oferują ogromne możliwości, kluczową obawą pozostaje prywatność danych. Jeśli modele korzystają z Twoich danych do douczania lub dostrajania, należy powstrzymać się od wysyłania wrażliwych informacji.

W następnym rozdziale poznasz umiejętności niezbędne do stosowania w praktyce funkcjonalnych modeli językowych, takich jak GPT-4. Odkrywanie potencjału generatywnej sztucznej inteligencji z pewnością przyniesie Ci wiele ekscytujących obserwacji, jak również praktycznych umiejętności.

Skorowidz

A

agenty, 217

- autonomiczne, 215
- dostosowywanie, 236
- funkcji OpenAI z pamięcią, 247
- rodzaje, 229
- schemat ReAct, 221
- stosowanie narzędzi, 227
- typu planuj i uruchamiaj, 249
- własne w LCEL, 237
- zestawy narzędzi, 234

algorytm word2vec, 189

analiza sentymentu, sentiment analysis, 104

- ograniczenia, 106
- usprawnienia, 105

API, 35, 228

architektura

- agenta BabyAGI, 249
- agentowa, agent-based architecture, 217
- samoodpytująca, 208
- transformera, 55

AutoGPT, 87

AUTOMATIC1111, 309

- domalowywanie, 324
- interfejs rozszerzenia ControlNet, 328
- tryb Img2Img, 316

B

BERT, 55

biblioteka

- FAISS, 195
- Gensim, 189
- spaCy, 96
- tiktoken, 100

błądzenie losowe, 259

BPE, Byte-Pair Encoding, 52

C

ChatGPT

- analiza sentymentu, 104
- czas na przemyślenie, 113
- filtrowanie danych, 76
- generowanie
 - danych w formacie CSV, 80
 - danych w formacie JSON, 73
 - danych w formacie YAML, 75
 - list, 67
 - list hierarchicznych, 69
 - różnorodnych formatów, 80
 - sekwencyjne, 106
- nieprawidłowe dokumenty, 77
- proces uczenia, 59
- pytanie o kontekst, 84
- rozdzielenie memów, 294
- technika wewnętrznego monologu, 114
- techniki promptowania, 112
- unikanie halucynacji, 112
- zasady generowania tekstu, 67
- zliczanie tokenów, 102

Claude 2, 64

ControlNet, 325

- Depth, 330
- metoda Canny'ego, 329
- M-LSD, 331
- Normal, 330
- OpenPose, 331
- segmentacja, 333
- SoftEdge, 332
- w aplikacji automatic1111, 328

ConversationBufferMemory, 242
ConversationBufferWindowMemory, 245
ConversationSummaryBufferMemory, 246
ConversationSummaryMemory, 245
ConversationTokenBufferMemory, 246
CoT, chain-of-thought reasoning, 215
czas na przemyślenie, thinking time, 112

D

DALL-E, 260
 wmalowywanie, 286
dane
 nieustrukturyzowane, 158
 ustrukturyzowane, 159
dekoder GPT, 55
dekompozycja zadań, task decomposition, 45, 166
detekcja krawędzi, 326
domalowywanie, outpainting, 288, 322
dopalamce jakości, quality boosters, 276
dostrajanie
 DreamBooth, 336
 modelu ChatGPT, 60
DreamBooth, 336
 uczenie modelu, 338, 342
drzewo myśli, ToT, 250
duże modele językowe, LLM, 17, 52
dzielenie na fragmenty, chunking, 186

E

edycja promptów, prompt editing, 315
enkoder BERT, 55
ewaluacje, evals, evaluations, 34, 138

F

FAISS, 181
 pozyskiwanie z pamięci, 195
filtrowanie dokumentów YAML, 76
Flask, 107
 dodawanie testów, 108
 planowanie architektury, 107
 programowanie funkcji, 107
format
 CSV, 80
 JSON, 73
 YAML, 75

framework
 Flask, 107
 LangChain, 126
 ReAct, 219
 ToT, 250
funkcja itemgetter, 170
funkcje
 asynchroniczne, 132
 OpenAI, 145, 228, 232
 przypadki użycia, 233

G

generowanie
 danych w formacie
 CSV, 80
 JSON, 73
 YAML, 75
 filmów, 268
 list, 67
 hierarchicznych, 69
 obrazów, 32, 257, 361
 z Midjourney, 270
 z opisów produktów, 122
 za pomocą Stable Diffusion, 303
 promptów stylu/cech, 123
 różnorodnych formatów, 80
 tekstu, 52, 355
 probabilistyczne, 55
 z ChatGPT, 67
 za pomocą LangChain, 126
 wzbogacone pozyskiwaniem, RAG, 183
 głosowanie większością, majority vote, 118
Google Gemini, 61, 268
GPT, Generative Pretrained Transformer, 55, 56
GPT -3.5-turbo, 59
GPT -4, 61
GPT -4V, 64

H

halucynacje, 11, 112

I

Img2Img, 316, 322
interfejs użytkownika, 366
Interrogate CLIP, 322
inżynieria promptów, prompt engineering, 11, 17

J

język LCEL, 132

K

klasyfikacja, 116
klucz API, 35
kodowania, 101
kodowanie parami bajtów, BPE, 52
kwantyzacja, 63

L

LangChain, 126
 ekstrakcja danych, 151
 ewaluacje, 138
 generowanie tekstu, 126
 język wyrażen, 132
 konfiguracja środowiska, 128
 ładowanie dokumentów, 193
 ładowarki dokumentów, 160
 łańcuchy
 promptów, 168
 sekwencyjne, 168
 łączenie danych, 158
 mapowanie i redukcja dokumentów, 178
 obsługa pamięci, 241
 parsery wyjścia, 134
 podział tekstu, 163
 rekursywny, 164
RAG, 199
rodzaje pamięci, 242, 245
rozdzielacze tekstu, 162
stosowanie PromptTemplate, 134
szablony promptów, 131
 z kilkoma przykładami, 153
wywołania zwrotne, 251–254
wywołanie funkcji, 149
zapisywanie i wczytywanie promptów, 157
zliczanie tokenów, 254
LCEL
 własne agenty, 237
Llama, 62
LLM, large language models, 17, 52
LoRA, 63

Ł

ładowanie dokumentów, document loading, 186
ładowarki dokumentów, 159, 160
łańcuch
 AI, AI chaining, 49
 myśli, chain-of-thought reasoning, 215
łańcuchowanie dokumentów, 179
łańcuchy
 dokumentów, 175
 LCEL, 175
 mapowania i redukcji, 178
 mapowania z rankingiem, 179
 nadziania dokumentów, 177
 oczyszczania dokumentów, 178
 promptów, 168
 sekwencyjne, 168

M

mapowanie memów, 298
mechanizm
 DreamBooth, 336
 podejmowania decyzji, 55
 uwagi, attention, 57
metaprompting, 47, 122, 292, 362
Midjourney, 262
 analiza promptów, 300
 domalowywanie, 288
 dopłacacze jakości, 276
 generowanie obrazów, 270
 inżynieria odwrotna promptów, 276
modyfikatory
 formatu, 270
 stylu w sztuce, 274
pojęcia ważne, 281
promptowanie z użyciem obrazków, 284
prompty negatywne, 278
przepisywanie promptów, 292
spójne postaci, 290
Mistral, 63
model
 ChatGPT, 59
 Claude 2, 64
 Google Gemini, 61
 GPT, 58
 GPT-3.5-turbo, 59
 GPT-4, 61

GPT-4V, 64
Llama, 62
Mistral, 63
segmentowania wszystkiego, SAM, 334
modele
 czatowe, 129
 strumieniowanie, 130
 szablon PromptTemplate, 134
 dyfuzyjne, diffusion models, 84, 257
 DALL-E, 260
 Midjourney, 262
 Stable Diffusion, 265
LLM, 17, 52
 ekstrakcja cech tekstu, 89
 ewaluacja kryteriów, 119
 generowanie streszczeń, 89
 generowanie tekstu, 52
 generowanie wielu odpowiedzi, 131
 głosowanie większością, 118
 jako API, 228
 łączenie podsumowań, 90
 podział tekstu na fragmenty, 92
 samooceniające odpowiedzi, 115
 tworzenie modelu klasyfikacji, 117
 uniwersalne tłumaczenia, 82
 zapisywanie i wczytywanie promptów, 157
 zasady użycia pamięci, 239
multimodalne, 64
osadzeń tekstów, 160

N

NLP, natural language processing, 56

O

ocena jakości odpowiedzi, 34, 212
oddalanie, zoom out, 288
odszumianie, 257, 316
okno przesuwne, 99
określenie
 formatu odpowiedzi, 28
 wytycznych, 23
OpenAI, 35
 DALL-E, 260
 modele, 58–61, 64
 wywołanie funkcji, 145
orkiestracja związków kontekstowych, 54
osadzenia, embeddings, 185, 259
 słów, word embeddings, 53

P

pakiety Pythona, 349
pamięć
 długoterminowa, 239
 krótkoterminowa, 240
parsery wyjścia, output parsers, 134
Pinecone, 181, 201
pisanie bloga, 348
 badanie tematu, 349
 generowanie grafiki, 361
 generowanie tekstu, 355
 generowanie zarysu posta, 353
 interfejs użytkownika, 366
 optymalizacja tytułu, 360
 reguły, 348
 styl, 357
 wywiad ekspercki, 352
planowanie zapytań, query planning, 151
pliki
 .csv, 37, 80
 .env, 35
 .json, 73
 .yml, 73, 76
podobieństwo kosinusowe, 191, 359
podział tekstu na fragmenty, chunking, 92
 biblioteka tiktoken, 100
 niewłaściwe przykłady, 93
 scenariusze, 93
 strategie, 95
 w Pythonie, 97
 za pomocą okna przesuwnego, 98
 zalety, 92
pozyskiwanie
 danych
 całościowe, 213
 kompresja kontekstowa, 213
 MultiQueryRetriever, 212
 samoodpytywanie, 208
 ważone czasem, 213
 z dokumentu źródłowego, 213
 informacji, 160
proces kodowania i dekodowania, 258
promptowanie
 chaotyczne, blind prompting, 34
 ekstrakcja cech tekstu, 89
 format odpowiedzi, 28
 inżynieria odwrotna, 276
 łańcuch promptów, 106, 168

- promptowanie
 - ocena jakości, 34
 - od najmniejszych do największych, 106
 - optymalizowanie promptów, 123
 - podział problemu, 45
 - przekazanie przykładów, 30
 - pytanie o kontekst, 84
 - techniki, 112
 - wydzielenie stylu tekstu, 87
 - wytyczne, 23
 - z użyciem obrazków, 284
 - z użyciem ról, 109–111
 - zasady, 17, 20
 - znajdowanie cech tekstu, 87
 - rozumienie dla pięciolatka, 81
- prompty, 17
 - negatywne, 278
 - z kilkoma przykładami
 - formatowanie, 154
 - o stałej długości, 153
 - ograniczenia, 157
 - szablony, 153
 - wybór według długości, 155
- przekazanie przykładów, 30
- przestrzeń ukryta, latent space, 182, 258
- przetwarzanie języka naturalnego, NLP, 56
- Python, 12

R

- RAG, Retrieval Augmented Generation, 184
 - użycie frameworka LangChain, 199
- ReAct, Reason and Act, 219, 232
 - implementacja schematu, 221
 - przypadki użycia, 234
- rozdzielacze tekstu, 162
- rozdzielenie, unbundling, 295
- rozgrzewanie, prewarming, 24
- rozkład zadania, task decomposition, 112

S

- SAM, Segment Anything Model, 334
- samoodpytywanie, 208
- silnik wnioskowania, reasoning engine, 76
- siła odsumiania, denoising strength, 316
- spaCy, 96
- Stable Diffusion, 265
 - ControlNet, 325
 - detekcja krawędzi Canny’ego, 326

- domalowywanie, 322
- generowanie obrazów, 303
- interfejs
 - SD Upscale, 320
 - webowy AUTOMATIC1111, 309
- mechanizm DreamBooth, 336
- skalowanie obrazków, 319
- tryb Interrogate CLIP, 322
- uruchamianie modelu, 303
- wmalowywanie, 322
- XL, 343
- strumieniowanie, streaming, 130
- szablony promptów, prompt templates, 131, 132
 - PromptTemplate, 134
 - z kilkoma przykładami, 153

T

- techniki
 - analiza sentymentu, 104
 - CoT, 215
 - czas na przemyślenie, 113
 - dekompozycja zadań, 166
 - dzielenie na fragmenty, 186
 - ekstrakcja cech tekstu, 89
 - kwantyzacja, 63
 - LoRA, 63
 - ładowanie dokumentów, 193
 - łańcuchy sekwencyjne, 168
 - metapromptowanie, 122
 - od najmniejszych do największych, 106, 108
 - podsumowywanie, 90
 - podział tekstu na fragmenty, 92
 - samooceniająca odpowiedź, 115
 - samoodpytywanie, 208
 - strumieniowanie, 130
 - TF-IDF, 191
 - ToT, 250
 - wewnętrzny monolog, 114
 - wydzielenie stylu tekstu, 87
- tekst wzorcowy, reference text, 89
- tensor, 58
- token, 52
- tokenizacja, 52, 101
- ToT, The Tree of Thoughts, 250
- TPU, Tensor Processing Unit, 58
- transformery, 54
 - dokumentów, 159
 - generatywne, 56

twarda przerwa, hard break, 281
tworzenie aplikacji, 348

U

uczenie
 bez przykładów, zero-shot learning, 117
 z kilkoma przykładami, few-shot learning, 117
usługa
 Google Colab, 339
 Pinecone, 201

W

wektor, 53, 181
wektorowa baza danych,
 vector database, 159, 181
 w chmurze, 201
wewnętrzne pozyskanie, internal retrieval, 24
wmalowywanie, inpainting, 286, 322
wnioskowanie na podstawie łańcucha myśli,
 CoT, 215
wnioskuj i działaj, ReAct, 219, 232
współbieżne wywołanie funkcji, 148

wydzielenie stylu tekstu,
 text style unbundling, 87
wykrywanie zdań, sentence detection, 96
wyrażenia
 LangChain, 132
 LCEL, 175
 regularne, 70, 222
wywołania zwrotne LangChain, 251
wywołanie funkcji, function calling, 145
 OpenAI, 145
 w LangChain, 149
 współbieżne, 148

Z

zamiana promptów, prompt switching, 315
zasady
 generowania tekstu, 67
 promptowania, 17, 20
zdolność
 rozumowania, reasoning capability, 84

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 

Absolutnie najlepsza książka na temat inżynierii promptów!

Dan Shipper, współzałożyciel i prezes Every

Udostępnienie dużych modeli językowych (ang. LLM, *large language model*) i modeli dyfuzyjnych, takich jak ChatGPT, Midjourney czy Stable Diffusion, zrewolucjonizowało wiele branż. Dzięki nim możliwa stała się realizacja szerokiego zakresu zadań, nawet tych, które do niedawna wydawały się niemożliwe do automatyzacji. Ponadto próg wejścia w świat AI jest bardzo niski, co sprawia, że niemal każdy może korzystać z modeli AI zgodnie ze swoimi potrzebami.

Z tą książką opanujesz podstawy generatywnej AI i nauczysz się skutecznie stosować jej modele w praktyce. Szczególną uwagę poświęcono integracji modeli językowych i dyfuzyjnych, co często bywa wyzwaniem, zwłaszcza w zakresie stabilności uzyskanych rozwiązań. Autorzy w jasny sposób wyjaśniają, jak za sprawą inżynierii promptów zapewnić niezawodność działania sztucznej inteligencji w środowiskach produkcyjnych. Co więcej, zaproponowane zasady są skonstruowane tak, aby bez trudu przetrwały próbę czasu i mogły być używane również dla przyszłych modeli!

Jeśli chcesz poprawić wiarygodność swoich systemów AI, musisz mieć tę książkę!

Mayo Oshin, założyciel i prezes Siennai Analytics

W książce:

- pięć uniwersalnych i perspektywicznych reguł promptowania
- korzystanie z generatywnej AI za pomocą bibliotek i frameworków, takich jak LangChain
- zalety i wady różnych modeli, w tym autorstwa OpenAI, i ich alternatyw
- praktyczne pisanie wysokiej jakości promptów w obszarze generowania tekstu, kodu i obrazów

James Phoenix tworzy stabilne potoki danych dla zespołów marketingowych, automatyzując tysiące powtarzalnych zadań. Wykładał *data science* w ramach ponad 60 bootcampów.

Mike Taylor współtworzył agencję marketingu wzrostowego Ladder. Prowadził niezwykle popularne kursy marketingu i AI w serwisach LinkedIn Learning, Udemy oraz Vexpower.

Obaj autorzy eksperymentowali z inżynierią promptów od 2020 roku, teraz pracując jako inżynierowie promptów.

Helion 

 helion.pl

 **HELION S.A.**
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶



ISBN 978-83-289-1904-4



9 788328 919044

Cena: 99,00 zł