Autonomiczna AI w praktyce

Jak budować inteligentne systemy zdolne do rozumowania, planowania i adaptacji

Anjanava Biswas
Wrick Talukdar

Helion

packt

Tytuł oryginału: Building Agentic AI Systems: Create intelligent, autonomous AI agents that can reason, plan, and adapt

Tłumaczenie: Krzysztof Sawka

ISBN: 978-83-289-3319-4

First published in the English language under the title 'Building Agentic AI Systems – (9781803238753)'.

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/auaipr
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: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

O autorach **13**

O recenzentach **16**

Przedmowa **17**

Przedmowa 2 ..... **19**

Wstęp **21**

CZĘŚĆ 1. Podstawy generatywnej sztucznej inteligencji i systemów agentowych

ROZDZIAŁ 1
Podstawy generatywnej sztucznej inteligencji **27**

Wprowadzenie do generatywnej sztucznej inteligencji 27
Rodzaje modeli generatywnej sztucznej inteligencji 29
Autokodery wariacyjne 29
Sieci GAN 30
Modele autoregresyjne i architektura transformera 31
Agenty AI oparte na modelach językowych 35
Zastosowania generatywnej sztucznej inteligencji 38
Wyzwania i ograniczenia generatywnej sztucznej inteligencji 44
Jakość danych i stronniczość 44
Prywatność danych 44
Zasoby obliczeniowe 45
Implikacje etyczne i społeczne 46
Uogólnianie i kreatywność 47
Podsumowanie 47
Pytania 47
Odpowiedzi 48
Dalsza lektura 48
Bibliografia 48

ROZDZIAŁ 2
Zasady systemów agentowych 50
Wymagania techniczne 50
Pojęcia samorządności, sprawczości i autonomii 50
Samorządność 51
Sprawczość 52
Autonomia 53
Przykład sprawczości i autonomii w agentach 54
Przegląd inteligentnych agentów i ich cech charakterystycznych 57
Architektura systemów agentowych 58
Architektury deliberatywne 59
Architektury reaktywne 60
Architektury hybrydowe 62
Systemy wieloagentowe 63
Definicja i cechy systemów wieloagentowych 65
Mechanizmy interakcji w systemach wieloagentowych 66
Podsumowanie 71
Pytania 72
Odpowiedzi 72

ROZDZIAŁ 3
Główne składniki inteligentnych agentów 73
Wymagania techniczne 73
Reprezentacja wiedzy w inteligentnych agentach 74
Sieci semantyczne 74
Ramki 76
Reprezentacje oparte na logice 77
Rozumowanie w inteligentnych agentach 78
Rozumowanie dedukcyjne 79
Rozumowanie indukcyjne 80
Rozumowanie abdukcyjne 82
Mechanizmy uczenia się agentów adaptacyjnych 83
Podejmowanie decyzji i planowanie w systemach agentowych 85
Funkcja użyteczności 85
Algorytmy planowania 88

Rozszerzanie możliwości agentów za pomocą generatywnej sztucznej inteligencji ... 91
Początek tworzenia agentowej sztucznej inteligencji ... 92
Podsumowanie ... 95
Pytania ... 95
Odpowiedzi ... 95

CZĘŚĆ 2. Projektowanie i wdrażanie agentów opartych na generatywnej sztucznej inteligencji

ROZDZIAŁ 4
Refleksja i introspekcja w agentach ... 99
Wymagania techniczne ... 100
Istota refleksji w agentach ... 100
Usprawnione podejmowanie decyzji ... 100
Adaptacja ... 101
Kwestie etyczne ... 102
Interakcja człowiek – komputer ... 103
Introspekcja w inteligentnych agentach ... 104
Implementacja możliwości refleksyjnych ... 105
Wnioskowanie tradycyjne ... 105
Metarozumowanie ... 105
Samowyjaśnianie ... 114
Samomodelowanie ... 118
Zastosowania i przykłady ... 120
Czatboty obsługi klienta ... 121
Osobiste agenty marketingowe ... 122
Systemy transakcji finansowych ... 123
Agenty prognozujące ... 124
Strategie cenowe w handlu elektronicznym ... 125
Podsumowanie ... 126
Pytania ... 127
Odpowiedzi ... 127

ROZDZIAŁ 5
Umożliwianie korzystania z narzędzi i planowania w agentach 129
Wymagania techniczne 130
Koncepcja wykorzystania narzędzi przez agenty 130
Wywoływanie funkcji i narzędzi 131
Definiowanie narzędzi dla agentów 133
Rodzaje narzędzi 134
Znaczenie narzędzi w systemach agentowych 137
Algorytmy planowania 138
Mniej praktyczne algorytmy planowania 139
Umiarkowanie praktyczny algorytm planowania: FF 140
Najbardziej praktyczne algorytmy planowania 141
Integracja wykorzystania narzędzi i planowania 147
Rozumowanie o narzędziach 148
Planowanie wykorzystania narzędzi 149
Analiza praktycznych zastosowań 150
Przykład z frameworkiem CrewAI 151
Przykład z frameworkiem AutoGen 152
Przykład z frameworkiem LangGraph 154
Podsumowanie 155
Pytania 156
Odpowiedzi 156

ROZDZIAŁ 6
Analiza podejścia koordynator – wykonawca – delegator 158
Wymagania techniczne 158
Model CWD 159
Kluczowe zasady modelu CWD 160
Model CWD inteligentnego agenta turystycznego 161
Projektowanie agentów z przypisanymi rolami 164
Role i obowiązki poszczególnych agentów 166
Komunikacja i współpraca agentów 171
Komunikacja 172
Mechanizm koordynacji 172
Negocjacje i rozwiązywanie konfliktów 172
Dzielenie się wiedzą 173

Wdrażanie podejścia CWD w systemach generatywnej sztucznej inteligencji ... 174
Prompty systemowe i zachowanie agenta ... 174
Formatowanie instrukcji ... 175
Wzorce interakcji ... 176
Podsumowanie ... 177
Pytania ... 177
Odpowiedzi ... 178

ROZDZIAŁ 7
Skuteczne techniki projektowania systemów agentowych ... 179
Wymagania techniczne ... 179
Ukierunkowane prompty systemowe i instrukcje dla agentów ... 180
Określanie celów ... 180
Specyfikacje zadań ... 181
Świadomość kontekstowa ... 183
Przestrzenie stanów i modelowanie środowiska ... 184
Reprezentacja przestrzeni stanów ... 185
Modelowanie środowiska ... 186
Wzorce integracji i interakcji ... 188
Monitorowanie i adaptacja ... 189
Architektura pamięci agenta i zarządzanie kontekstem ... 190
Pamięć krótkotrwała (pamięć robocza) ... 190
Pamięć długotrwała (baza wiedzy) ... 192
Pamięć epizodyczna (historia interakcji) ... 193
Zarządzanie kontekstem ... 194
Integracja z procesem podejmowania decyzji ... 195
Przetwarzanie sekwencyjne i przetwarzanie równoległe w agentowych cyklach pracy ... 196
Przetwarzanie sekwencyjne ... 196
Przetwarzanie równoległe ... 197
Optymalizacja cyklu pracy ... 198
Podsumowanie ... 200
Pytania ... 200
Odpowiedzi ... 201

CZĘŚĆ 3. Zaufanie, bezpieczeństwo, etyka i zastosowania

ROZDZIAŁ 8
Budowanie zaufania do systemów generatywnej sztucznej inteligencji 205
Wymagania techniczne 206
Znaczenie zaufania w dziedzinie sztucznej inteligencji 206
Metody budowania zaufania 207
Przejrzystość i wyjaśnialność 207
Radzenie sobie z niepewnością i uprzedzeniami 211
Skuteczne komunikowanie wyników 213
Kontrola i zgoda ze strony użytkownika 214
Etyka i odpowiedzialność 215
Wdrażanie przejrzystości i wyjaśnialności 216
Radzenie sobie z niepewnością i uprzedzeniami 217
Podsumowanie 219
Pytania 219
Odpowiedzi 219

ROZDZIAŁ 9
Zarządzanie kwestiami bezpieczeństwa i etyki 222
Potencjalne zagrożenia i wyzwania 222
Ataki adwersarialne 223
Uprzedzenia i dyskryminacja 224
Dezinformacja i halucynacje 225
Naruszenia prywatności danych 227
Zagrożenia związane z własnością intelektualną 228
Przygotowanie bezpiecznej i odpowiedzialnej AI 230
Standardy i ramy etyczne 234
Projektowanie zorientowane na człowieka 234
Odpowiedzialność 234
Prywatność i ochrona danych 235
Zaangażowanie różnych interesariuszy 235
Rozwiązywanie problemów z zakresu prywatności i bezpieczeństwa 236
Podsumowanie 237
Pytania 238
Odpowiedzi 238

ROZDZIAŁ 10
Typowe przypadki użycia i zastosowania ... 240
Zastosowania kreatywne i artystyczne ... 241
Ewolucja agentów kreatywnych i artystycznych ... 241
Zastosowania w praktyce ... 241
Przetwarzanie języka naturalnego i agenty konwersacyjne ... 244
Ewolucja agentów językowych ... 244
Zastosowania w praktyce ... 245
Robotyka i systemy autonomiczne ... 247
Ewolucja agentów robotycznych ... 247
Zastosowania w praktyce ... 247
Wspomaganie decyzji i optymalizacja ... 250
Ewolucja systemów wspomagania decyzji ... 250
Zastosowania w praktyce ... 251
Podsumowanie ... 254
Pytania ... 254
Odpowiedzi ... 255

ROZDZIAŁ 11
Podsumowanie i perspektywy na przyszłość ... 257
Podsumowanie kluczowych koncepcji ... 258
Nowe trendy i kierunki badań ... 259
Inteligencja multimodalna, czyli integracja różnorodnych danych wejściowych ... 259
Zaawansowane rozumienie języka ... 259
Uczenie się przez doświadczenie, czyli innowacje w uczeniu przez wzmacnianie ... 260
Praktyczne zastosowania w różnych branżach ... 260
Ogólna sztuczna inteligencja ... 261
Co wyróżnia ogólną sztuczną inteligencję? ... 261
Największe wyzwanie ... 261
Nauka uczenia się ... 262
Zrozumienie rzeczywistego świata ... 262
Wyzwania i możliwości ... 263
Podsumowanie ... 265

Skorowidz ... 267

Rozdział 2

Zasady systemów agentowych

W poprzednim rozdziale przybliżyliśmy podstawy generatywnej sztucznej inteligencji, omówiliśmy rodzaje modeli generatywnych oraz pokrótce przedstawiliśmy agenty AI oparte na dużych modelach językowych. W tym rozdziale zajmiemy się podstawowymi zasadami systemów agentowych, począwszy od krótkiego omówienia pojęć sprawczości i autonomii, a następnie przechodząc do charakterystyki inteligentnych agentów. Omówimy także różne architektury systemów agentowych oraz systemy wieloagentowe, posługując się zaprezentowanym w poprzednim rozdziale przykładem asystenta do rezerwacji podróży.

Oto główne tematy tego rozdziału:

- pojęcia samorządności, sprawczości i autonomii;
- przegląd inteligentnych agentów i ich cech charakterystycznych;
- analiza architektury systemów agentowych;
- omówienie systemów wieloagentowych.

Po ukończeniu tego rozdziału będziesz mieć opanowane ogólne podstawy inteligentnych agentów oraz najważniejsze aspekty architektury systemów agentowych, które należy wziąć pod uwagę przy budowie inteligentnego systemu agentowego.

Wymagania techniczne

Pliki z kodem źródłowym do tego rozdziału znajdziesz w repozytorium na GitHubie pod adresem *https://github.com/PacktPublishing/Building-Agentic-AI-Systems*. Aby skonfigurować swoje środowisko programistyczne, postępuj zgodnie z instrukcjami zawartymi w pliku *README*.

Pojęcia samorządności, sprawczości i autonomii

Fascynującym aspektem **systemów agentowych** (ang. *agentic system*) są wykorzystywane przez nie złożone procesy decyzyjne. Dostarczają one cennych informacji na temat optymalizacji wyborów w określonych kontekstach. Systemy te często kwestionują nasze konwencjonalne rozumienie odpowiedzialności.

Systemy agentowe są siłą napędową innowacji i postępu technologicznego w różnych dziedzinach, takich jak robotyka, sztuczna inteligencja i inżynieria systemów. Ich rozwój i wdrażanie przyczyniły się do eksploracji i tworzenia nowych form automatyzacji i inteligentnych zachowań. Przyjrzyjmy się niektórym obszarom, w których systemy agentowe zyskują na znaczeniu:

- **Robotyka**. W dziedzinie robotyki systemy agentowe utorowały drogę do projektowania i wdrażania autonomicznych robotów zdolnych do poruszania się w złożonych środowiskach, wykonywania skomplikowanych zadań i dostosowywania się do zmiennych warunków. Roboty te, wyposażone w zdolności decyzyjne i sprawczość, znalazły zastosowanie m.in. w produkcji, eksploracji, akcjach poszukiwawczo-ratowniczych i opiece zdrowotnej. Na przykład roboty z zachowaniami agentowymi mogą samodzielnie poruszać się po terenach dotkniętych katastrofą, oceniać potencjalne zagrożenia i podejmować decyzje wspierające akcje ratunkowe, wykazując się inteligentnym i adaptacyjnym zachowaniem.
- **Sztuczna inteligencja**. W dziedzinie AI systemy agentowe odegrały kluczową rolę w rozwoju inteligentnych agentów i systemów wspomagania decyzji. Systemy te wykorzystują zaawansowane algorytmy, techniki uczenia maszynowego i metody reprezentacji wiedzy do analizowania danych, wnioskowania w złożonych scenariuszach oraz dostarczania inteligentnych rekomendacji lub zautomatyzowanych możliwości podejmowania decyzji. Agentowe systemy AI znalazły zastosowanie w takich dziedzinach jak finanse, ochrona zdrowia, transport i marketing, umożliwiając bardziej efektywne procesy decyzyjne.
- **Inżynieria systemów**. W inżynierii systemów systemy agentowe ułatwiły projektowanie i wdrażanie złożonych, rozproszonych i adaptacyjnych systemów. Systemy te często składają się z wielu współdziałających komponentów lub podsystemów, z których każdy wykazuje zachowania agentowe i zdolności decyzyjne. Takie rozwiązania znajdują zastosowanie m.in. w sieciach energetycznych, sieciach transportowych i systemach cyberfizycznych, gdzie inteligentne i autonomiczne podejmowanie decyzji jest kluczowe dla efektywnego działania, alokacji zasobów i odporności na awarie.

Kluczowe dla koncepcji systemów agentowych są pojęcia samorządności, sprawczości i autonomii. Omówimy każde z tych pojęć w kolejnych punktach, aby zrozumieć ich istotę i poznać kluczową rolę, jaką odgrywają w architekturze systemów agentowych.

Samorządność

Systemy agentowe to sztuczne i ludzkie systemy posiadające zdolność do samorządności, adaptacji i interakcji. Samorządność odnosi się do zdolności systemu lub bytu do samodzielnego zarządzania sobą, bez zewnętrznego kierowania czy kontroli. W kontekście systemów agentowych **samorządność** (ang. *self-governance*) oznacza, że system może podejmować własne decyzje, wyznaczać własne cele i regulować swoje zachowanie w oparciu o wewnętrzne reguły, modele i algorytmy decyzyjne. W skrócie: działają one zgodnie z własnymi regułami i stanami wewnętrznymi, a w razie potrzeby zmieniają swoje zachowanie w odpowiedzi na zmiany w otoczeniu lub celach. Takie systemy wchodzą

w interakcje ze środowiskiem lub innymi systemami w sposób dla nich istotny i poprzez te interakcje ulegają wpływom.

Oto kluczowe aspekty samorządności w systemach agentowych:

- **Samoorganizacja**. Zdolność organizowania i strukturyzowania własnych procesów wewnętrznych, zasobów i zachowań bez ingerencji z zewnątrz.
- **Samoregulacja**. Umiejętność monitorowania i dostosowywania własnych działań i wyników na podstawie informacji zwrotnych ze środowiska lub stanów wewnętrznych, aby zapewnić działanie w ramach pożądanych parametrów lub ograniczeń.
- **Samoadaptacja**. Zdolność modyfikowania swojego zachowania, strategii lub procesów decyzyjnych w odpowiedzi na zmiany w otoczeniu lub własnych warunkach wewnętrznych, aby skuteczniej osiągać swoje cele.
- **Samooptymalizacja**. Zdolność ciągłego doskonalenia swojej wydajności, efektywności lub możliwości podejmowania decyzji poprzez uczenie się, zdobywanie doświadczeń lub procesy ewolucyjne.
- **Samostanowienie**. Zdolność wyznaczania własnych celów, priorytetów i kierunków działania w oparciu o wewnętrzne procesy decyzyjne, bez całkowitego uzależnienia od sił zewnętrznych.

Samorządność w systemach agentowych jest często uzyskiwana poprzez integrację różnych technologii, struktur i metod, takich jak uczenie maszynowe, reprezentacja wiedzy, wnioskowanie i algorytmy decyzyjne. Pozwalają one systemowi przetwarzać informacje, uczyć się na podstawie danych i doświadczeń oraz podejmować autonomiczne decyzje w oparciu o zdobytą wiedzę i aktualny kontekst.

Sprawczość

Sprawczość (ang. *agency*) to zdolność jednostki lub innego podmiotu do niezależnego działania i podejmowania decyzji. W kontekście systemów ludzkich i sztucznych sprawczość obejmuje następujące kluczowe elementy:

- **Autonomia decyzyjna**. Odnosi się do możliwości działania i wykonywania czynności zgodnie z wybraną opcją lub planem działania. Systemy posiadające sprawczość mają swobodę oceny różnych możliwości i wyboru najodpowiedniejszego działania na podstawie własnych procesów decyzyjnych, a nie tylko w oparciu o zewnętrzne czynniki czy z góry ustalone reguły.
- **Intencjonalność**. Sprawczość zakłada istnienie intencji, celów lub zamierzeń, które kierują działaniami i zachowaniem systemu. Systemy sprawcze nie są jedynie reaktywne — mają poczucie celu i potrafią dążyć do konkretnych założeń, dostosowując swoje działania i strategie w miarę potrzeby, aby te cele osiągnąć.
- **Odpowiedzialność**. Sprawczość jest ściśle powiązana z pojęciem odpowiedzialności, czyli koniecznością rozliczenia się z wyników i konsekwencji własnych działań. Systemy posiadające sprawczość uznaje się za odpowiedzialne za podejmowane decyzje i wpływ ich działań na otoczenie lub inne podmioty, z którymi wchodzą w interakcje.

W większości przypadków sprawczość sztucznej inteligencji polega na zdolności systemu do autonomicznego podejmowania decyzji w oparciu o wewnętrzne programy, modele i przetwarzane dane. Decyzje te mogą mieć istotny wpływ na funkcjonowanie samego systemu lub jego interakcje z otoczeniem.

Wróćmy do przedstawionego w poprzednim rozdziale przykładu asystenta, który rezerwuje bilety lotnicze, a być może nawet noclegi w hotelach. W tym przypadku system wykazałby się sprawczością, analizując różne czynniki, takie jak dostępność lotów między dwoma miastami, ceny biletów oraz wszelkie inne ograniczenia podane przez użytkownika, jak preferowana klasa miejsc itp. Na tej podstawie podejmowałby decyzje o optymalnym wyszukiwaniu lotów i hoteli spełniających te kryteria, aby zminimalizować ogólny koszt podróży. System byłby odpowiedzialny za rezultaty swoich decyzji, które wpłynęłyby na ogólny plan podróży i jej koszt dla klienta.

Autonomia

Autonomia (ang. *autonomy*) jest ściśle powiązana z pojęciem sprawczości, ale koncentruje się bardziej na stopniu niezależności, jaką posiada dany podmiot lub system. Można ją podzielić na kilka aspektów:

- **Autonomia operacyjna**. Określa zdolność systemu do wykonywania określonego zadania lub zestawu zadań bez bezpośredniej ingerencji lub kontroli ze strony człowieka. System z autonomią operacyjną może samodzielnie realizować swoje funkcje, opierając się na własnych procesach wewnętrznych, algorytmach decyzyjnych i zdolności odczytywania otoczenia.
- **Autonomia funkcjonalna**. Ten aspekt autonomii dotyczy zdolności systemu do dokonywania wyborów i podejmowania działań w celu osiągnięcia wyznaczonych celów z uwzględnieniem środowiska lub kontekstu, w którym działa. Systemy funkcjonalnie autonomiczne potrafią dostosowywać swoje zachowanie i procesy decyzyjne w odpowiedzi na zmieniające się warunki lub bodźce, co pozwala im skuteczniej realizować swoje cele.
- **Autonomia hierarchiczna**. Ten aspekt odnosi się do zakresu uprawnień decyzyjnych przyznanych systemowi w ramach szerszej struktury lub organizacji. Systemy o wyższej autonomii hierarchicznej mają większą swobodę w podejmowaniu decyzji wpływających na ich podsystemy lub szersze działania, natomiast systemy o niższej autonomii hierarchicznej mogą podlegać większym ograniczeniom lub nadzorowi ze strony podmiotów wyższego szczebla.

W dziedzinie sztucznej inteligencji i robotyki autonomia jest kluczowym pojęciem, definiującym stopień, w jakim system może wykonywać zadania i podejmować decyzje bez konieczności ciągłej ingerencji człowieka.

W przykładzie naszego asystenta rezerwacji lotów system posiadałby autonomię operacyjną do wykonywania takich zadań jak rezerwacja lotów czy noclegów, zarządzanie przypomnieniami i pozyskiwanie informacji o podróży przy minimalnym lub zerowym udziale człowieka. Posiadałby również autonomię funkcjonalną, pozwalającą mu interpretować polecenia użytkownika, dostosowywać się do indywidualnych preferencji i podejmować decyzje zgodne z celami i kontekstem użytkownika. Poziom autonomii

hierarchicznej przyznany takiemu systemowi mógłby zależeć od czynników takich jak preferencje użytkownika dotyczące prywatności czy dostęp systemu do wrażliwych danych i zasobów.

Warto zauważyć, że autonomia w systemach sztucznej inteligencji i robotycznych nie oznacza koniecznie całkowitego braku nadzoru lub kontroli ze strony człowieka. Często systemy te działają w ramach ściśle określonych granic i ograniczeń ustalonych przez ich projektantów lub operatorów, zachowując jednocześnie znaczny stopień autonomii. W naszym przykładzie asystenta rezerwacji lotów czatbot prosi użytkownika o dodatkowe informacje, takie jak daty podróży, lokalizacje, imię i nazwisko oraz adres, aby dokonać rezerwacji lotu. Następnie porównuje te dane z dostępnymi lotami, sugerując opcje odpowiadające preferencjom użytkownika. Jeśli jakiś szczegół jest niejasny lub niedostępny, czatbot prosi użytkownika o wyjaśnienie, zapewniając dokładność, a jednocześnie działając autonomicznie w granicach określonych przez swoje oprogramowanie.

Pojęcia samorządności, sprawczości i autonomii w systemach sztucznej inteligencji często wiążą się z kwestiami etycznymi, szczególnie w odniesieniu do poziomu autonomii przyznawanej tym systemom oraz potencjalnych zagrożeń i konsekwencji ich decyzji. W miarę jak systemy sztucznej inteligencji stają się coraz bardziej zaawansowane i zdolne do samodzielnego podejmowania decyzji, kluczowe staje się zapewnienie ich zgodności z ludzkimi wartościami i zasadami etycznymi.

Przykład sprawczości i autonomii w agentach

Zilustrujmy koncepcję sprawczości i autonomii na przykładzie prostego algorytmu asystenta rezerwacji podróży. Warto zauważyć, że ten algorytm niekoniecznie wykorzystuje sztuczną inteligencję, ale pomoże zrozumieć omawiane pojęcia. Algorytm asystenta rezerwacji podróży może wyglądać tak jak w ramce.

Algorytm 1. Algorytm asystenta rezerwacji podróży ze sprawczością i autonomią

Wymaga: Nazwy agenta N

Zapewnia: Zainicjalizowany obiekt A TravelAgent ze sprawczością i autonomią

```
Initialize A ← CreateTravelAgent(N)
Initialize A.goals ← empty list
Initialize A.knowledge_base ← empty dictionary

// Sprawczość: Zdolność do działania w imieniu użytkownika
function SetGoal(G)
A. goals.Append(G) // Sprawczość: Definiowanie celów
function UpdateKnowledge(K, V)
A.knowledge_base[K] ← V // Sprawczość: Pozyskiwanie informacji z API
i ocenianie
```

```
// Autonomia: Zdolność do samodzielnego działania
8. function MakeDecision(Options)
9. best_option ← max(Options, key =score)  // Autonomia: Niezależne
   podejmowanie decyzji
10. return best_option
11. function BookTravel(Departure, Destination)
12. Output „Agent A.name is booking travel to Destination"

// Sprawczość: Wykonanie działania w imieniu użytkownika
13. SetGoal(„Book flight from Departure to Destination")
14. UpdateKnowledge({Departure, Destination})

// Autonomia: Samodzielne zarezerwowanie podróży poprzez znalezienie
   najlepszego lotu
15. MakeDecision()

// Tutaj zaimplementuj logikę rezerwacji i zapisz w A
16. Output A
```

Oto objaśnienie działania tego algorytmu demonstrującego zdolności sprawczości i autonomii:

1. Zaczynamy od nadania nazwy agentowi — nazwijmy go `TripPlanner`.
2. Następnie inicjalizujemy nowy obiekt `TravelAgent` z nazwą `N = "TripPlanner"` — reprezentuje to utworzenie jednostki zdolnej zarówno do sprawczości, jak i autonomii.
3. Teraz tworzymy listę do przechowywania celów agenta. Odnosi się to do sprawczości, ponieważ cele reprezentują intencje lub pożądane rezultaty, do których agent będzie dążył w imieniu użytkownika. Jest to oznaczone jako `A.goals ← empty list`.
4. Inicjalizujemy pusty słownik (`empty dictionary`; znany również jako mapa lub pary klucz – wartość) do przechowywania wiedzy agenta. Jest to kluczowe zarówno dla sprawczości (działanie w imieniu użytkowników), jak i autonomii (niezależne działanie), ponieważ będzie zawierać informacje, których agent używa do podejmowania decyzji.
5. Kroki 4. i 5. w algorytmie wskazują definicję funkcji, która dodaje nowy cel `G` do listy celów agenta. Przypomina to przyjmowanie przez agenta celów w imieniu użytkownika. Jest to oznaczone jako `A.goals.Append(G)`. Można to traktować jako fragment kodu, który odbierze wiadomość od użytkownika, na przykład: „Zarezerwuj mi lot z San Diego do Seattle". W tym przypadku celem jest zarezerwowanie lotu z San Diego do Seattle.
6. Kroki 6. i 7. w algorytmie wskazują definicję funkcji, która aktualizuje bazę wiedzy agenta o nową parę klucz – wartość (mapę lub słownik). Reprezentuje to sprawczość poprzez pozyskiwanie informacji, które będą wykorzystywane

do działania w imieniu użytkownika. Wspiera również autonomię, dostarczając agentowi informacji, których może użyć do podejmowania niezależnych decyzji. Ta operacja jest reprezentowana jako `A.knowledge_base[K] ← V`. W naszym przypadku funkcja ta wykorzystuje (teoretycznie) kilka API związanych z podróżami, aby uzyskać opcje lotów między dwoma miastami, wprowadzając w ten sposób wiedzę do bazy wiedzy. Jest to również miejsce, w którym każda z tych opcji lotów zostanie oceniona, na przykład późne loty otrzymują niskie oceny, a wczesne loty otrzymują wyższe oceny.

7. Kroki od 8. do 10. definiują funkcję, która wykonuje kilka różnych rzeczy. Przyjmuje listę opcji i wybiera najlepszą na podstawie pewnych kryteriów oceny. Jest to przykład autonomii w algorytmie, ponieważ agent niezależnie ocenia opcje i podejmuje decyzję bez bezpośredniej ingerencji człowieka.
8. Kroki od 10. do 15. pokazują, jak wszystkie te komponenty współpracują ze sobą, począwszy od ustalenia celu rezerwacji lotu z wykorzystaniem miast wylotu i przylotu, poprzez aktualizację bazy wiedzy za pomocą API wyszukiwania lotów, aż po ocenę dostępnych połączeń. Następnie wykorzystuje funkcję `MakeDecision` do znalezienia najlepszego możliwego lotu na podstawie najwyższego wyniku i dokonuje rezerwacji dla użytkownika.

Poniższy fragment kodu przedstawia implementację funkcji `BookTravel` w języku Python:

```
1. def book_travel(departure: str, destination: str):
2.     self.set_goal(f"Book flight from {departure} to {destination}")
3.     self.update_knowledge(departure, destination)
4.
5.     try:
6.        best_flight = self.make_decision()
7.        booking_confirmation = f"BOOKING_#12345"
8.        self.knowledge_base['booking_confirmation'] = \
9.                          booking_confirmation
10.       print(f"Rezerwacja potwierdzona: {booking_confirmation}")
11.    except Exception as e:
12.       print(f"Rezerwacja zakończona niepowodzeniem: {str(e)}")
13.
14. if __name__ == "__main__":
15.    agent = TravelAgent("TripPlanner")
16.    agent.book_travel("SAN", "SEA")
17.    print("\n----------Końcowy stan agenta: -----------")
18.    print(f"Nazwa: {agent.name}")
19.    print(f"Cele: {agent.goals}")
20.    if 'booking_confirmation' in agent.knowledge_base:
21.      print(f"Potwierdzenie rezerwacji: \
22.      {agent.knowledge_base['booking_confirmation']}")
```

Wynik działania tego kodu, gdy agent zostanie zainicjalizowany w celu zarezerwowania lotu z SAN (San Diego) do SEA (Seattle), wygląda następująco:

```
1. Agent TripPlanner is booking travel from SAN to SEA
2. Goal set: Book flight from SAN to SEA
3. Knowledge updated with 3 flight options
4. Decision made: Selected flight JetBlue
5. Rezerwacja potwierdzona: BOOK-JetBlue-TRIPPLANNER
```

```
6. ---------- Końcowy stan agenta: ----------
7. Nazwa: TripPlanner
8. Cele: [Book flight from SAN to SEA']
9. Potwierdzenie rezerwacji: BOOK-JetBlue-TRIPPLANNER
```

Pełną implementację agenta planującego podróż można znaleźć w notatniku Pythona *Chapter_02.ipynb* w repozytorium na GitHubie.

W tym fragmencie kodu funkcja `book_travel` przyjmuje kod miasta wylotu (np. `SAN` lub `SEA`, które są kodami lotnisk), po czym wywołuje inne funkcje, aby ustawić cel, zaktualizować bazę wiedzy, a następnie podjąć decyzję o wyborze lotu i dokonać rezerwacji. Warto zauważyć, że nasz agent, mimo pewnej funkcjonalności i autonomii, nie jest inteligentny. Nie potrafi odczytać wiadomości tekstowych od użytkownika i zrozumieć jego intencji, aby ustawić cele, zaktualizować bazę wiedzy i wykonać odpowiednie działania, lecz wymaga podania kodów lotnisk. Jednak jak widzieliśmy w przykładzie, użytkownik (lub klient) może wyrazić swoje intencje w prostym języku, mówiąc na przykład: „Zarezerwuj mi lot z San Diego do Seattle”.

W obecnej formie wprowadzanych przez użytkownika danych (komunikatu) agent nie jest w stanie określić, jakie są miasta wylotu i przylotu, czego użytkownik oczekuje, ani nawet co oznacza wprowadzony ciąg tekstu. To właśnie w tym miejscu wkracza generatywna sztuczna inteligencja, o czym przekonamy się w kolejnych rozdziałach. Na razie kontynuujmy omówienie, przyglądając się cechom charakterystycznym agentów.

Przegląd inteligentnych agentów i ich cech charakterystycznych

Inteligentny agent to złożony, samorządny byt, który postrzega swoje otoczenie i podejmuje działania w celu zrealizowania określonych celów lub zadań. Agenty te mogą być zarówno prostymi systemami ściśle przestrzegającymi predefiniowanych reguł, jak i zaawansowanymi systemami zdolnymi do uczenia się i adaptacji na podstawie doświadczeń. Oto kluczowe cechy inteligentnych agentów:

- **Reaktywność**. Agenty reaktywne reagują w czasie rzeczywistym na zmiany i zdarzenia zachodzące w ich otoczeniu. Nieustannie monitorują swoje środowisko i dostosowują do niego swoje zachowanie. Ta reaktywność pozwala agentom dostosowywać się do dynamicznych warunków i odpowiednio reagować na bodźce, dzięki czemu ich działania są adekwatne i skuteczne.
- **Proaktywność**. Idealny inteligentny agent nie powinien jedynie reagować na zdarzenia, ale również wykazywać zachowania proaktywne. Agenty proaktywne przewidują przyszłe potrzeby, wyzwania lub możliwości i z własnej inicjatywy planują oraz działają odpowiednio do sytuacji. Są zorientowane na cel i aktywnie realizują strategie prowadzące do jego osiągnięcia, zamiast po prostu reagować na pojawiające się okoliczności.
- **Zdolności społeczne**. Wiele inteligentnych agentów działa w systemach wieloagentowych, gdzie współpracują z innymi agentami lub ludźmi, aby osiągnąć wspólne cele wymagające zbiorowego wysiłku. Zdolności społeczne

obejmują umiejętności komunikacji, koordynacji i negocjacji, umożliwiające agentom efektywną współpracę i wykorzystanie zbiorowej inteligencji lub zasobów.

Dzięki tym kluczowym cechom inteligentne agenty wykazują niezwykłą wszechstronność i efektywność w szerokim spektrum dziedzin i scenariuszy. Ich możliwości pozwalają im doskonale radzić sobie z zadaniami od prostych, zautomatyzowanych procesów po wysoce złożone, dynamiczne sytuacje decyzyjne wymagające adaptacji w czasie rzeczywistym i reagowania na zmiany w środowisku. Oprócz tych podstawowych cech inteligentne agenty mogą mieć również inne zaawansowane umiejętności:

- **Uczenie się i adaptacja**. Inteligentne agenty mają zdolność uczenia się na podstawie doświadczeń i dostosowywania swojego zachowania w czasie. Mogą zdobywać nową wiedzę, udoskonalać procesy decyzyjne i poprawiać swoje działanie dzięki technikom takim jak uczenie maszynowe, uczenie przez wzmacnianie czy algorytmy ewolucyjne.
- **Rozumowanie i planowanie**. Inteligentne agenty mogą wykorzystywać zdolności rozumowania i planowania do analizy złożonych sytuacji, formułowania strategii i podejmowania świadomych decyzji. Mogą korzystać z technik takich jak reprezentacja wiedzy, wnioskowanie logiczne i algorytmy planowania, aby poruszać się w skomplikowanych przestrzeniach problemowych i określać optymalne sposoby działania.
- **Autonomia i samorządność**. Inteligentne agenty często wykazują pewien stopień autonomii i samorządności, pozwalający im podejmować decyzje i działać niezależnie, bez ciągłej ingerencji czy nadzoru ze strony człowieka. Ta autonomia umożliwia agentom efektywne funkcjonowanie w dynamicznych środowiskach lub scenariuszach, gdzie ciągła kontrola ludzka jest niepraktyczna lub niemożliwa.

Dzięki tym cechom inteligentne agenty mogą być wszechstronne i efektywne w wielu dziedzinach, od prostych, zautomatyzowanych zadań po wysoce złożone, dynamiczne sytuacje decyzyjne. Znajdują zastosowanie w takich obszarach jak robotyka, systemy wspomagania decyzji, wirtualne asystenty, gry czy symulacje.

Architektura systemów agentowych

Systemy agentowe, zaprojektowane do autonomicznego realizowania złożonych celów, mogą być implementowane przy użyciu rozmaitych wzorców architektonicznych. Ogólnie rzecz biorąc, wzorce te określają strukturę i zachowanie, pozwalające systemowi skutecznie postrzegać, rozumować, uczyć się i oddziaływać z otoczeniem. Trzy główne wzorce architektoniczne dla systemów agentowych to architektury deliberatywne, reaktywne i hybrydowe. Przyjrzyjmy się im bliżej.

Architektury deliberatywne

Architektury **deliberatywne** (ang. *deliberative*), znane również jako architektury **oparte na wiedzy** (ang. *knowledge-based*) lub **symboliczne** (ang. *symbolic*), polegają na wykorzystaniu jawnych reprezentacji wiedzy i mechanizmów wnioskowania do podejmowania decyzji. Zazwyczaj działają one według cyklu **postrzeganie – planowanie – działanie** (ang. *sense-plan-act*), gdzie najpierw odbierają informacje o środowisku, następnie tworzą plan działania na podstawie tych spostrzeżeń i bazy wiedzy, a na końcu wykonują zaplanowane czynności.

Główną zaletą architektur deliberatywnych jest ich zdolność radzenia sobie z zadaniami wymagającymi złożonego rozumowania, takimi jak planowanie, rozwiązywanie problemów i podejmowanie decyzji. Architektury te wykorzystują techniki takie jak wnioskowanie oparte na regułach, spełnialność więzów i przeszukiwanie heurystyczne do poruszania się w skomplikowanych przestrzeniach problemowych i formułowania odpowiednich planów działania.

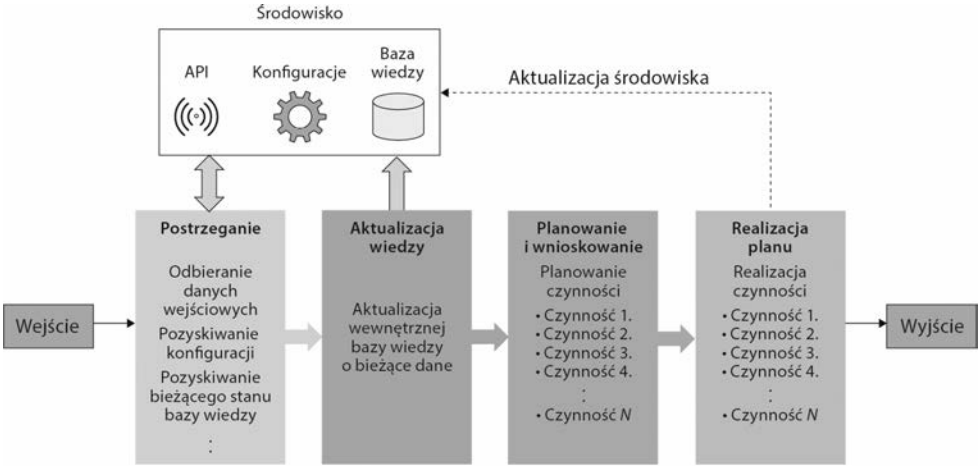
Jednym z kluczowych elementów architektury deliberatywnej jest baza wiedzy, która przechowuje symboliczne reprezentacje środowiska, celów, ograniczeń i wiedzy dziedzinowej. Baza ta jest zazwyczaj zakodowana przy użyciu języka formalnego lub logiki, co umożliwia systemowi przeprowadzanie wnioskowania logicznego. Cykl *postrzeganie – planowanie – działanie* w architekturach deliberatywnych obejmuje zwykle następujące etapy:

1. **Postrzeganie**. Agent odbiera i gromadzi informacje o środowisku za pomocą różnych czujników lub mechanizmów wejściowych.
2. **Aktualizacja wiedzy**. Zebrane informacje są wykorzystywane do aktualizacji wewnętrznej bazy wiedzy agenta, zapewniając utrzymanie dokładnej reprezentacji aktualnego stanu środowiska.
3. **Planowanie i wnioskowanie**. Na podstawie zaktualizowanej bazy wiedzy agent stosuje techniki wnioskowania i algorytmy do formułowania planów i podejmowania decyzji. Może to obejmować techniki takie jak spełnialność więzów, wnioskowanie logiczne, algorytmy przeszukiwania lub planowanie heurystyczne.
4. **Realizacja planu**. Po ustaleniu planu lub sposobu działania agent wykonuje odpowiednie czynności w środowisku, potencjalnie modyfikując je lub osiągając określone cele.

Rysunek 2.1 przedstawia architekturę deliberatywną systemu agentowego z cyklem postrzeganie – planowanie – działanie:

Architektury deliberatywne doskonale radzą sobie z zadaniami wymagającymi złożonego rozumowania, planowania i podejmowania decyzji w dobrze zdefiniowanych środowiskach. Mogą one skutecznie radzić sobie z niepewnością i niejednoznacznością dzięki technikom takim jak wnioskowanie probabilistyczne, logika rozmyta czy mechanizmy rewizji przekonań.

Jednak architektury deliberatywne mają również pewne wady. Jednym z istotnych wyzwań jest koszt obliczeniowy związany z utrzymywaniem i wnioskowaniem na podstawie



Rysunek 2.1. Architektura deliberatywna systemu agentowego

złożonych baz wiedzy, co może ograniczać responsywność w czasie rzeczywistym w dynamicznych środowiskach. Ponadto jawna reprezentacja wiedzy może być problematyczna w dziedzinach, w których wiedzę trudno sformalizować lub jest ona stale ewoluująca.

Aby zaradzić tym ograniczeniom, architektury deliberatywne są często łączone z komponentami reaktywnymi lub behawioralnymi w architekturach hybrydowych. Pozwala to na złożone rozumowanie, a jednocześnie szybkie reagowanie na zmiany w otoczeniu.

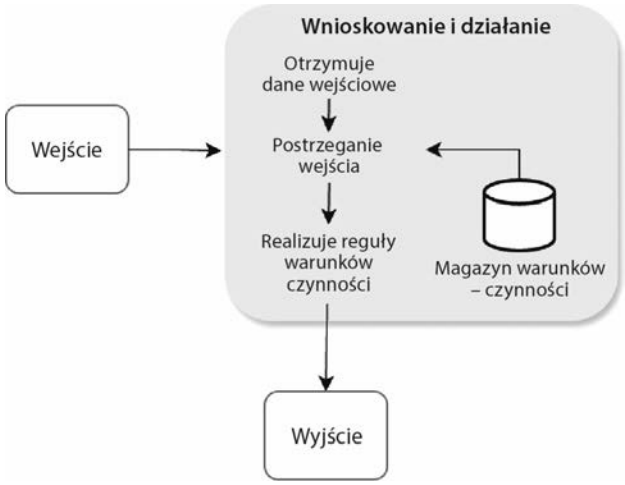
Mimo swoich ograniczeń architektury deliberatywne pozostają kluczowym elementem wielu inteligentnych systemów, szczególnie w dziedzinach wymagających złożonego podejmowania decyzji, planowania i wnioskowania, takich jak robotyka, systemy wspomagania decyzji czy inteligentne systemy nauczania.

Architektury reaktywne

Architektury **reaktywne** (ang. *reactive*), znane również jako architektury **oparte na zachowaniach** (ang. *behawior-based*) lub **bodźcach i reakcjach** (ang. *stimulus-response*), mają na celu zapewnienie natychmiastowych odpowiedzi na bodźce z otoczenia. W przeciwieństwie do architektur deliberatywnych architektury reaktywne nie opierają się na jawnych modelach świata ani złożonych procesach rozumowania, lecz bezpośrednio odzwierciedlają postrzeganie na działania, zazwyczaj wykorzystując proste reguły warunkowe lub sieci neuronowe, co przedstawiliśmy na rysunku 2.2.

Oto niektóre z kluczowych właściwości i cech architektur reaktywnych:

- **Szybkość i responsywność**. Architektury reaktywne są zaprojektowane tak, aby błyskawicznie reagować na zmiany w otoczeniu. Dzięki bezpośredniemu powiązaniu spostrzeżeń z działaniami mogą one pomijać czasochłonne procesy rozumowania, umożliwiając szybkie i terminowe reakcje.



Rysunek 2.2. Architektura reaktywna systemu agentowego

- **Odporność i tolerancja na błędy**. Te architektury są zasadniczo odporne i mniej podatne na zakłócenia lub niekompletne informacje. Ich prosta, autonomiczna natura sprawia, że są mniej narażone na katastrofalne awarie, ponieważ poszczególne elementy lub zachowania mogą kompensować lub łagodzić skutki błędnych lub brakujących danych wejściowych, szczególnie gdy są używane w ramach architektury deliberatywnej.
- **Radzenie sobie z niepewnością**. Architektury reaktywne potrafią skutecznie radzić sobie z niepewnością w dynamicznych środowiskach. Ich zdolność bezpośredniego reagowania na bodźce środowiskowe pozwala im adaptować i dostosowywać działania do bieżącej sytuacji, bez polegania na precyzyjnych lub kompletnych modelach całego świata.
- **Przetwarzanie równoległe i rozproszone**. Architektury reaktywne często wykorzystują przetwarzanie równoległe i rozproszone, używając wielu modułów reaktywnych, które działają jednocześnie i niezależnie. To zdecentralizowane podejście umożliwia efektywne radzenie sobie ze złożonymi zadaniami oraz zapewnia naturalną skalowalność i modułowość.
- **Skomplikowane zachowania**. Mimo prostoty poszczególnych zachowań lub reguł interakcja i koordynacja wielu komponentów reaktywnych może prowadzić do wyłaniania się złożonych, przypominających inteligencję zachowań na poziomie całego systemu.

Choć architektury reaktywne oferują zalety w zakresie szybkości, odporności i radzenia sobie z niepewnością, mają też pewne ograniczenia:

- **Brak planowania długoterminowego**. Architektury reaktywne zasadniczo nie mają zdolności planowania z wyprzedzeniem lub rozważania długoterminowych konsekwencji. Ich skupienie na natychmiastowych reakcjach na bodźce środowiskowe utrudnia realizację złożonych, wieloetapowych celów lub strategii.
- **Ograniczone rozumowanie i abstrakcja**. Te architektury mogą mieć trudności z zadaniami wymagającymi abstrakcyjnego rozumowania, uogólniania lub

operowania reprezentacjami symbolicznymi. Są one przede wszystkim zaprojektowane do działania na niższym poziomie bodźców i reakcji.

- **Ograniczone możliwości uczenia się**. Wiele architektur reaktywnych nie ma zdolności uczenia się na podstawie doświadczeń ani dostosowywania swojego zachowania w czasie. Ich sztywny zestaw reguł lub zachowań może nie być odpowiedni dla dynamicznych środowisk lub zadań wymagających ciągłego uczenia się i adaptacji.

Mimo tych ograniczeń architektury reaktywne są szeroko stosowane w dziedzinach, w których kluczowe są responsywność w czasie rzeczywistym, odporność i umiejętność radzenia sobie z niepewnością, takich jak robotyka, gry wideo wykorzystujące sztuczną inteligencję czy systemy sterowania. Ponadto architektury reaktywne często służą jako składniki bardziej złożonych architektur hybrydowych, uzupełniając systemy deliberatywne lub oparte na uczeniu się, aby osiągnąć pożądany poziom wydajności i adaptowalności.

Architektury hybrydowe

Badacze dostrzegli zalety i ograniczenia zarówno architektur deliberatywnych, jak i reaktywnych, co doprowadziło do rozwoju architektur hybrydowych, mających na celu wykorzystanie zalet obu podejść. Takie architektury hybrydowe zazwyczaj stosują strukturę warstwową:

- **Warstwa reaktywna** (ang. *reactive layer*) do szybkich, doraźnych reakcji. Warstwa ta odpowiada za obsługę interakcji z otoczeniem w czasie rzeczywistym, zapewniając szybkie i adekwatne do sytuacji reakcje na bodźce zewnętrzne. Jest ona zaprojektowana tak, aby była wysoce responsywna, odporna na błędy i zdolna do radzenia sobie z niepewnością, wykorzystując mocne strony architektur reaktywnych.
- **Warstwa deliberatywna** (ang. *deliberative layer*) do ogólnego wnioskowania i planowania. Warstwa ta jest przeznaczona do bardziej zaawansowanych procesów rozumowania, planowania i podejmowania decyzji. Może ona utrzymywać bardziej kompleksową reprezentację środowiska, celów i ograniczeń, umożliwiając formułowanie złożonych strategii, rozumowanie o abstrakcyjnych pojęciach i planowanie długoterminowych działań.

Interakcja między tymi dwiema warstwami jest kluczowa dla umożliwienia systemom agentowym skutecznego reagowania na dynamiczne konteksty środowiskowe przy jednoczesnym zachowaniu zdolności planowania działań i rozumowania w związku z nimi. Warstwa reaktywna może dostarczać warstwie deliberatywnej informacje zwrotne w czasie rzeczywistym i zapewniać świadomość sytuacyjną, wpływając na jej procesy decyzyjne. Z kolei warstwa deliberatywna może kierować i wpływać na zachowanie warstwy reaktywnej, dostarczając ogólne plany, cele i ograniczenia.

Aby osiągnąć złożone cele i wykorzystać mocne strony obu warstw, w architekturach hybrydowych często są stosowane następujące techniki:

- **Dekompozycja zadań**. Rozbijanie złożonych zadań na podzadania, które mogą być realizowane przez odpowiednią warstwę. Warstwa reaktywna zajmuje się

zadaniami szczegółowymi, krytycznymi czasowo, a warstwa deliberatywna skupia się na ogólniejszym planowaniu i koordynacji.

- **Opracowywanie wielu planów**. Warstwa deliberatywna może generować wiele potencjalnych planów lub strategii, a warstwa reaktywna może dynamicznie wybierać i wykonywać najbardziej odpowiedni wariant na podstawie aktualnych warunków środowiskowych.
- **Planowanie z wykorzystaniem modułów zewnętrznych**. Warstwa deliberatywna może wykorzystywać zewnętrzne moduły lub specjalistyczne algorytmy do zadań takich jak planowanie ścieżki, alokacja zasobów czy harmonogramowanie, korzystając z wiedzy i technik specyficznych dla danej dziedziny.
- **Refleksja i doskonalenie**. Warstwa deliberatywna może analizować wyniki wykonanych planów, uczyć się na podstawie doświadczeń i dopracowywać swoje procesy rozumowania i planowania, umożliwiając ciągłe doskonalenie i adaptację.
- **Planowanie wspomagane pamięcią**. Warstwa deliberatywna może utrzymywać pamięć lub historię przeszłych doświadczeń, decyzji i wyników, umożliwiając wykorzystanie tej wiedzy w przyszłych procesach planowania i rozumowania.

Dzięki zaletom podejść deliberatywnego i reaktywnego architektury hybrydowe dążą do zrównoważenia responsywności i rozumowania, co pozwala tworzyć bardziej odporne, autonomiczne i adaptowalne systemy agentowe. Architektury te wykorzystują mocne strony obu podejść, zapewniając możliwość szybkiego reagowania na dynamiczne środowiska przy jednoczesnym zachowaniu zdolności do złożonego planowania, wnioskowania i podejmowania decyzji.

Projektowanie i implementacja efektywnych architektur hybrydowych jest aktywnym obszarem badań. Naukowcy dążą do opracowania architektur, które mogą płynnie integrować i koordynować komponenty deliberatywne i reaktywne, by umożliwiać tworzenie wysoce wydajnych i inteligentnych systemów agentowych.

Wybór odpowiedniego wzorca architektonicznego dla systemu agentowego zależy od konkretnych wymagań aplikacji i uwzględnia takie czynniki jak złożoność zadań, niepewność środowiska i potrzeba responsywności w czasie rzeczywistym. Architektury deliberatywne sprawdzają się w scenariuszach wymagających złożonego rozumowania i procesów decyzyjnych, natomiast architektury reaktywne doskonale radzą sobie w dynamicznych środowiskach wymagających szybkich i adaptacyjnych reakcji. Architektury hybrydowe osiągają harmonijną równowagę, umiejętnie wykorzystując zalety obu paradygmatów, co prowadzi do tworzenia bardziej wydajnych i adaptowalnych systemów agentowych, które mogą płynnie poruszać się w złożonych środowiskach operacyjnych.

Systemy wieloagentowe

Systemy wieloagentowe (ang. *multi-agent system* — MAS) stanowią ważną dziedzinę szerszego obszaru rozproszonej sztucznej inteligencji. Składają się z kilku inteligentnych agentów, które wchodzą w interakcje, współpracują i koordynują swoje działania w celu wykonywania zadań i osiągania wspólnych celów. Każdy agent w systemie MAS jest

zazwyczaj autonomiczny, postrzega otoczenie za pomocą czujników, posiada mechanizm wnioskowania umożliwiający podejmowanie decyzji oraz działa na podstawie tych decyzji, aby spełnić swoje cele projektowe. Zbiorowe zachowanie i interakcje tych agentów pozwalają systemom MAS rozwiązywać złożone problemy, z którymi pojedyncze systemy agentowe mają trudności ze względu na ograniczenia poszczególnych agentów.

Przykłady systemów MAS można znaleźć w różnych dziedzinach, co dowodzi ich przydatności i skuteczności w rozwiązywaniu złożonych problemów:

- **Zarządzanie łańcuchem dostaw i logistyka**. Systemy MAS mogą być wykorzystywane do optymalizowania operacji w łańcuchu dostaw poprzez koordynowanie działań różnych agentów reprezentujących dostawców, producentów, dystrybutorów i sprzedawców detalicznych. Każdy agent może podejmować decyzje na podstawie swojej lokalnej wiedzy i ograniczeń, współpracując jednocześnie z innymi agentami w celu zapewnienia efektywnej alokacji zasobów, zarządzania zapasami i planowania transportu.
- **Kontrola ruchu i systemy transportowe**. Systemy MAS znalazły zastosowanie w zarządzaniu przepływem ruchu i optymalizacji sieci transportowych. Agenty mogą reprezentować poszczególne pojazdy, sygnalizację świetlną lub centra zarządzania ruchem, współpracując w celu zmniejszenia zatorów, koordynowania sygnalizacji świetlnej i znajdowania optymalnych tras dla pojazdów na podstawie aktualnych warunków drogowych.
- **Robotyka i produkcja**. W środowiskach produkcyjnych systemy MAS mogą koordynować działania wielu robotów lub zautomatyzowanych systemów. Każdy robot lub agent może być odpowiedzialny za konkretne zadania, takie jak montaż, spawanie lub obsługa materiałów, komunikując się i koordynując swoje działanie z innymi agentami w celu zapewnienia wydajnych i zsynchronizowanych operacji.
- **Monitorowanie środowiska i zarządzanie zasobami**. Systemy MAS mogą być wykorzystywane do monitorowania i zarządzania zasobami naturalnymi, takimi jak sieci wodociągowe, gospodarka leśna czy ochrona siedlisk dzikich zwierząt. Agenty mogą reprezentować różnych interesariuszy, czujniki środowiskowe lub podmioty decyzyjne, współpracując w celu podejmowania świadomych decyzji dotyczących alokacji zasobów, działań ochronnych lub strategii minimalizowania negatywnych skutków.
- **Rozproszone sieci czujników**. Systemy MAS są dobrze przystosowane do zastosowań związanych z rozproszonymi sieciami czujników, takich jak monitorowanie środowiska, nadzór czy reagowanie na katastrofy. Każdy węzeł czujnika może być reprezentowany jako agent, zbierający i przetwarzający lokalne dane, jednocześnie koordynując działania z innymi agentami w celu łączenia informacji i zapewnienia kompleksowego zrozumienia monitorowanego obszaru lub zjawiska.
- **Inteligentne środowiska wirtualne i symulacje**. Systemy MAS mogą być wykorzystywane do tworzenia inteligentnych środowisk wirtualnych i symulacji, w których agenty reprezentują różne podmioty lub aktorów w symulowanym świecie. Agenty te mogą wchodzić w interakcje, podejmować decyzje i wykazywać złożone zachowania, umożliwiając realistyczne symulacje systemów społecznych, modeli ekonomicznych czy operacji wojskowych.

Główne zalety systemów MAS tkwią w ich zdolności do rozproszenia możliwości rozwiązywania problemów, wykorzystania zbiorowej inteligencji i specjalizacji poszczególnych agentów oraz wykazywania odporności i tolerancji na błędy poprzez zdecentralizowane podejmowanie decyzji. Ponadto systemy MAS mogą ułatwiać integrację heterogenicznych komponentów, umożliwiając rozwój elastycznych i skalowalnych systemów zdolnych zajmować się złożonymi, dynamicznymi problemami, które byłyby trudne do rozwiązania przy użyciu monolitycznych, scentralizowanych podejść.

Definicja i cechy systemów wieloagentowych

System wieloagentowy składa się z wielu autonomicznych agentów, które mogą wchodzić w interakcje, współpracować i współdziałać po to, aby osiągnąć wspólne cele. Agentami mogą być programy komputerowe, roboty, a nawet ludzie wyposażeni w specjalistyczne umiejętności i cele. Interakcja między agentami jest niezbędnym elementem, umożliwiającym im efektywną współpracę, wymianę informacji i podział zadań w oparciu o ich mocne strony i obszary specjalizacji. Oto kluczowe cechy systemów wieloagentowych:

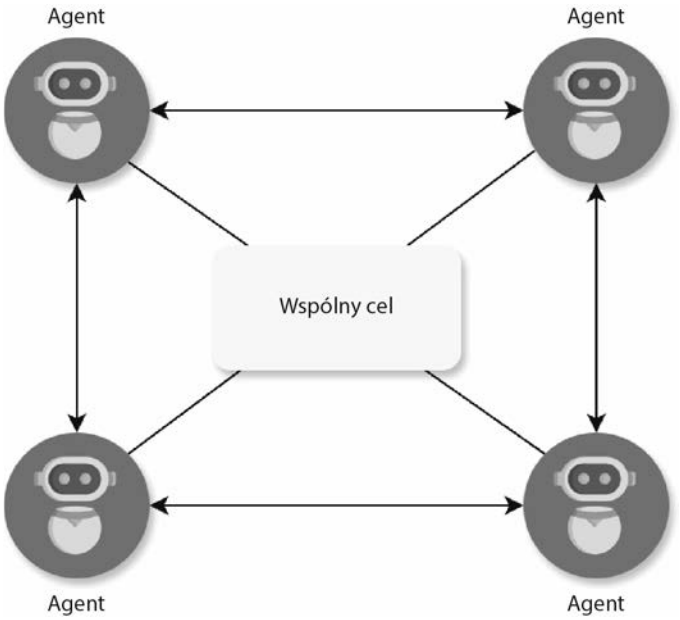
- **Autonomia**. Każdy agent w MAS jest samodzielny i podejmuje niezależne decyzje na podstawie swojego postrzegania środowiska i własnych celów. Agenty działają niezależnie, bez centralnej kontroli, wykazując autonomiczne zachowanie.
- **Interakcja**. Agenty w MAS komunikują się ze sobą za pomocą określonych protokołów, co pozwala im na wymianę informacji, negocjowanie zadań i koordynowanie działań. Interakcja ta może przybierać różne formy, takie jak współpraca, koordynacja lub rywalizacja, w zależności od charakteru problemu i celów agentów.
- **Adaptacyjność**. MAS mają zdolność dostosowywania się i zmiany zachowania w odpowiedzi na zmiany w środowisku lub zmiany celów poszczególnych agentów. Ta adaptacyjność sprawia, że systemy wieloagentowe są w stanie radzić sobie z dynamicznymi sytuacjami, co czyni je elastycznymi i odpornymi podczas działania.
- **Rozproszona kontrola**. W przeciwieństwie do systemów scentralizowanych MAS wykorzystują rozproszoną kontrolę, gdzie podejmowanie decyzji i sterowanie są rozdzielone między poszczególne agenty. Ta rozproszona kontrola przyczynia się do odporności systemu, ponieważ awarie lub nieprawidłowe działanie jednego agenta nie muszą wpływać na funkcjonalność całego systemu.
- **Skalowalność**. Architektura MAS jest z natury skalowalna, gdyż umożliwia dodawanie lub usuwanie agentów w zależności od potrzeb. Ta skalowalność pozwala systemowi modyfikować złożoność i możliwości, co czyni go odpowiednim dla szerokiego zakresu zastosowań.
- **Heterogeniczność**. Agenty w MAS mogą być heterogeniczne, co oznacza, że mogą mieć różne architektury, możliwości i cele. Ta niejednorodność pozwala na integrację różnorodnych komponentów i wykorzystanie specjalistycznej wiedzy, przyczyniając się do ogólnej efektywności systemu.
- **Zdecentralizowane dane i wiedza**. W MAS dane i wiedza są zdecentralizowane i rozproszone między poszczególnymi agentami. Ta decentralizacja zwiększa odporność, ponieważ nie ma pojedynczego punktu awarii, a agenty mogą działać na podstawie swojej lokalnej wiedzy i spostrzeżeń.

Zdolność MAS do rozproszenia możliwości rozwiązywania problemów, wykorzystania inteligencji zbiorowej, wykazywania odporności i integracji heterogenicznych komponentów sprawia, że są one dobrze przystosowane do rozwiązywania złożonych, dynamicznych problemów, które stanowią wyzwanie dla tradycyjnych, scentralizowanych podejść.

Mechanizmy interakcji w systemach wieloagentowych

Mechanizmy interakcji w systemach wieloagentowych odgrywają kluczową rolę w umożliwianiu efektywnej komunikacji, współpracy i koordynacji między agentami. Podstawowe mechanizmy interakcji w MAS można podzielić na trzy główne typy:

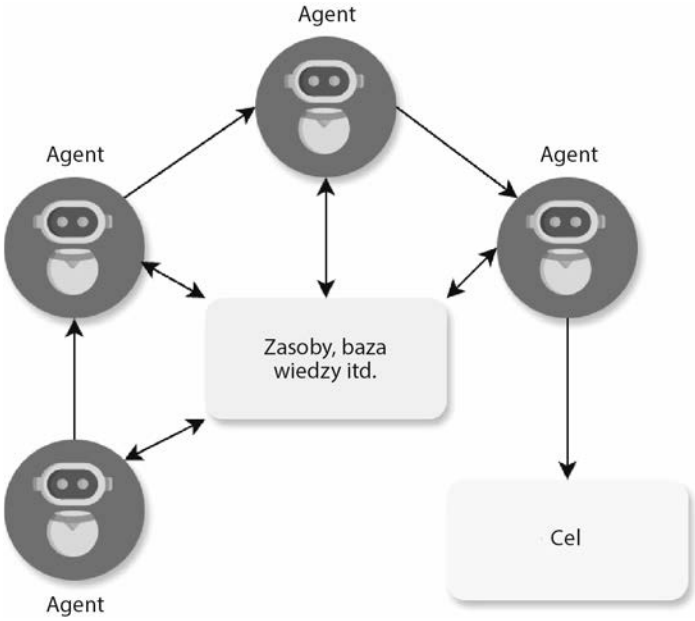
- **Współpraca**. Można ją zdefiniować jako współdziałanie agentów w celu osiągnięcia wspólnego celu. Jest ona szczególnie istotna w sytuacjach, gdy żaden pojedynczy agent działający samodzielnie nie jest w stanie zrealizować zadania (rysunek 2.3).



Rysunek 2.3. Współpraca w systemach wieloagentowych

Doskonałym przykładem współpracy w MAS są operacje ratunkowe podczas katastrof, gdzie wiele dronów, robotów i ludzi musi współdziałać, aby skutecznie lokalizować i ratować ofiary. MAS opiera się na współpracy agentów poprzez łączenie ich wiedzy, zasobów i wysiłków w celu realizacji zadań zbyt złożonych dla jednego agenta. Agenty mogą współpracować, dzieląc zadania, łącząc swoją specjalistyczną wiedzę lub uzupełniając swoje umiejętności, aby efektywniej rozwiązywać skomplikowane problemy.

- **Koordynacja**. Polega na zarządzaniu współzależnościami wynikającymi z działań i aktywności agentów w systemie. Jest ona niezbędna, gdy agenty dzielą zasoby i mają nakładające się obowiązki lub kolidujące działania (rysunek 2.4).



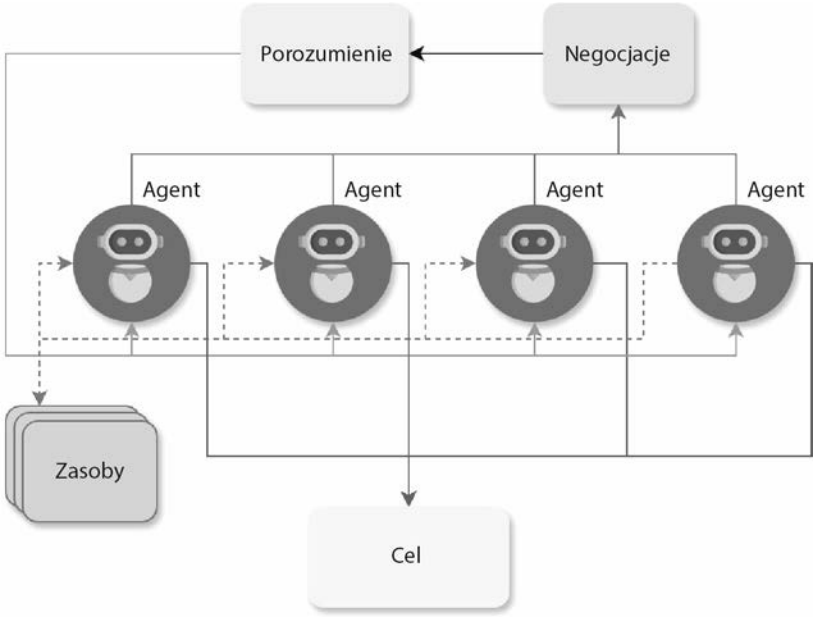
Rysunek 2.4. Koordynacja w systemach wieloagentowych

  Mechanizmy koordynacji w MAS mogą obejmować strategie takie jak harmonogramowanie zadań, zarządzanie alokacją zasobów i rozwiązywanie konfliktów. Na przykład w środowisku produkcyjnym agenty reprezentujące różne roboty na liniach produkcyjnych muszą koordynować swoje działania, aby zapewnić efektywne wykorzystanie wspólnych zasobów, zapobiegać zakłóceniom i utrzymać ogólną wydajność produkcji.

- **Negocjacje**. Jest to proces, w którym agenty dochodzą do porozumienia w sprawie podziału zasobów, rozdziału zadań lub rozwiązywania konfliktów. Obejmuje on składanie ofert, wysuwanie kontrpropozycji i zawieranie kompromisów, nawet gdy początkowe interesy agentów są sprzeczne (rysunek 2.5).

  Mechanizmy negocjacji w MAS umożliwiają agentom znalezienie wzajemnie korzystnych rozwiązań poprzez wymianę propozycji, ocenę alternatyw i osiąganie konsensusu. Jest to szczególnie przydatne w sytuacjach, gdy agenty mają ograniczone lub wykluczające się wzajemnie zasoby, różne preferencje lub konkurencyjne cele. Negocjacje mogą obejmować różne techniki, takie jak aukcje, protokoły głosowania, strategie targowania się lub podejścia oparte na teorii gier, w zależności od specyficznych wymagań i ograniczeń danej dziedziny problemu.

Te mechanizmy interakcji — współpraca, koordynacja i negocjacje — są fundamentalne dla efektywnego funkcjonowania MAS. Umożliwiają one agentom współpracę, wykorzystanie ich zbiorowych możliwości oraz rozwiązywanie konfliktów lub maksymalizowanie



Rysunek 2.5. Negocjacje w systemach wieloagentowych

korzyści płynących ze współzależności, które mogą pojawić się podczas ich interakcji. Wybór i projektowanie odpowiednich mechanizmów interakcji są kluczowe dla tworzenia wydajnych i odpornych systemów wieloagentowych, które mogą rozwiązywać złożone problemy i dostosowywać się do dynamicznych środowisk.

W kontekście naszego przykładu asystenta podróży, systemy wieloagentowe mogą odgrywać istotną rolę w ułatwianiu efektywnej koordynacji i negocjacji między różnymi podmiotami zaangażowanymi w sieć branży turystycznej. W takim scenariuszu agenty mogą reprezentować różnych interesariuszy, takich jak linie lotnicze, hotele, wypożyczalnie samochodów i biura podróży i wykorzystywać mechanizmy negocjacji do optymalizacji różnych aspektów rezerwowania podróży.

Na przykład rozważmy MAS, w którym agenty reprezentują linie lotnicze, hotele i inne istotne podmioty z branży turystycznej. Agenty te mogą realizować procesy negocjacyjne w celu ustalenia rozkładów lotów, dostępności pokoi, cen i innych decyzji związanych z podróżą, dążąc do osiągnięcia najwyższej efektywności rezerwacji podróży.

Proces negocjacji może przebiegać w następujący sposób:

1. Agenty linii lotniczych proponują dostępne miejsca, rozkłady lotów i ceny dla swoich tras.
2. Agenty hoteli oceniają te propozycje na podstawie dostępności pokoi, przewidywanego obłożenia i prognoz popytu, a następnie negocjują z agentami linii lotniczych najbardziej odpowiednie rozkłady lotów, które są zgodne z ich godzinami zameldowania i wymeldowania.

3. Agenty biur podróży negocjują zarówno z agentami linii lotniczych, jak i z agentami hoteli, biorąc pod uwagę preferencje klientów, ograniczenia budżetowe oraz konkretne wymagania dotyczące dat podróży i zakwaterowania.
4. Agenty firm transportowych (np. wypożyczalni samochodów lub firm oferujących przejazdy) również mogą uczestniczyć w procesie negocjacji, oferując usługi transportu naziemnego oraz proponując harmonogramy odbioru/dowozu i związane z tym koszty innym uczestniczącym agentom.

W trakcie procesu negocjacji agenty mogą wykorzystywać różne strategie i algorytmy do oceny propozycji, generowania kontrofert i znajdowania wzajemnie akceptowalnych porozumień. Strategie te mogą obejmować techniki takie jak aukcje, protokoły negocjacyjne, podejścia z teorii gier lub algorytmy optymalizacyjne dostosowane do działań w branży turystycznej. Na przykład agenty mogą stosować wieloatrybutowe funkcje użyteczności, które uwzględniają czynniki takie jak czas podróży, koszt, komfort i preferencje klientów, aby oceniać i tworzyć rankingi różnych propozycji. Następnie mogą przeprowadzać iteracyjne negocjacje, dostosowując swoje oferty i kontroferty na podstawie swoich funkcji użyteczności i ograniczeń.

Co więcej, rozproszony charakter systemów wieloagentowych umożliwia zdecentralizowane podejmowanie decyzji, gdzie każdy agent może dokonywać wyborów na podstawie swojej lokalnej wiedzy i ograniczeń, jednocześnie współpracując i koordynując działania z innymi agentami w celu osiągnięcia globalnych celów optymalizacyjnych. Mechanizmy negocjacyjne w systemach wieloagentowych dla naszego przykładu z branży turystycznej nie tylko ułatwiają efektywną koordynację między różnymi podmiotami, ale także zapewniają elastyczność i zdolność adaptacji do dynamicznych zmian w popycie, podaży, cenach lub innych czynnikach operacyjnych, co ostatecznie prowadzi do bardziej odpornego i responsywnego systemu, który zaspokaja potrzeby klientów.

Aby zilustrować system wieloagentowy w naszym przykładzie asystenta rezerwacji lotów, wprowadzimy kilka nowych funkcjonalności. Chcemy teraz, aby nasz system nie tylko rezerwował loty, ale także znajdował hotele w miejscu docelowym i tworzył odpowiedni pakiet podróży dla klienta. Algorytm dla takiego systemu wieloagentowego mógłby wyglądać tak jak w ramce.

Algorytm 2. System wieloagentowy dla asystenta rezerwacji podróży

Wymaga: Zestawu agentów lotniczych A = {A1, A2, ..., An} i agentów hotelowych H = {H1, H2, ..., Hm}

Zapewnia: Zainicjalizowany system rezerwacji podróży S z agentem biura podróży TA

```
1. Initialize S with A, H, and TA
2. function RequestTravelPackage(departure, destination, dates)
3.     for each Ai in A do
4.         available_flights ← Ai.GetAvailableFlights(departure, destination, dates)
5.     for each Hj in H do
6.         available_rooms ← Hj.GetAvailableRooms(destination, dates)
```

```
    packages ← TA.CompilePackages(available_flights, available_rooms)
    return packages
function BookTravel(selected_package)
    flight_booking ← selected_package.airline.BookFlight()
    room_booking ← selected_package.hotel.BookRoom()
    if flight_booking and room_booking are successful then
        return CreateBooking(flight_booking, room_booking)
    else
        return FailureNotification()
function UpdateDynamicPricing()
    for each Ai in A do
        Ai.UpdateFlightPrices()
    for each Hj in H do
        Hj.UpdateRoomPrices()
while True do
    if NewTravelRequest() then
        request ← GetTravelRequest()
        packages ← RequestTravelPackage(request.departure,
        request.destination, request.dates)
        selected_package ← TA.PresentOptionsToCustomer(packages)
        if selected_package is not null then
            booking ← BookTravel(selected_package)
            if booking is successful then
                NotifyCustomer(booking, „Rezerwacja potwierdzona")
            else
                NotifyCustomer („Rezerwacja nie powiodła się")
    if TimeToUpdatePricing() then
        UpdateDynamicPricing()
Output S
```

Oto przegląd kluczowych elementów tego algorytmu:

1. Pierwszym krokiem jest jasne zdefiniowanie zestawu agentów: w tym przypadku agenta linii lotniczych, agenta hoteli i agenta biura podróży. Agenty lotniczy i hotelowy odpowiadają za działania związane z lotami i noclegami, a agent biura podróży jest odpowiedzialny za tworzenie pakietów podróży na podstawie najlepszych dostępnych opcji.
2. Kroki od 2. do 8. pokazują, jak agent biura podróży współpracuje z wieloma agentami linii lotniczych i hoteli w celu utworzenia pakietów podróży. Znajduje odpowiednie rozkłady lotów i dostępność hoteli w mieście docelowym, a następnie wykorzystuje te dane do tworzenia pakietów.

3. Kroki od 9. do 15. przedstawiają koordynację między wybranymi agentami linii lotniczych i hoteli w celu potwierdzenia rezerwacji lotu i noclegu zgodnie z wybranym pakietem.
4. Kroki od 16. do 20. pokazują, jak każdy agent linii lotniczych i hoteli niezależnie aktualizuje swoje ceny.
5. Główna pętla od kroku 21. do 33. łączy wszystko razem, pokazując, jak system obsługuje zapytania o podróże i okresowo aktualizuje ceny dla wszystkich agentów.

Ten przykładowy algorytm demonstruje połączenie koordynacji i współpracy między agentami:

- współpraca, ponieważ wszystkie agenty pracują nad wspólnym celem, jakim jest zarezerwowanie planu podróży dla użytkownika;
- koordynacja, ponieważ agent biura podróży potrzebuje danych zarówno od agenta lotniczego, jak i od agenta hotelowego, aby utworzyć pakiet podróży, a następnie zarezerwować najlepszy pakiet.

Pełny kod w Pythonie związany z tym algorytmem można znaleźć w notatniku *Chapter_02.ipynb* w repozytorium na GitHubie. Warto pamiętać, że podobnie jak wcześniej nasz system wieloagentowy nie jest zbyt inteligentny, ponieważ do poprawnego działania nadal wymaga konkretnych danych wejściowych, takich jak kody miast wylotu i przylotu. Brakuje mu zdolności zrozumienia lub wnioskowania wartości i działań na podstawie wiadomości lub tekstu wprowadzonego przez użytkownika.

Podsumowanie

W tym rozdziale zgłębiliśmy fascynujący świat systemów agentowych i inteligentnych agentów, analizując kluczowe koncepcje sprawczości i autonomii oraz cechy charakteryzujące idealnego agenta. Przyjrzeliśmy się różnym wzorcom architektonicznym służącym do projektowania i implementowania takich systemów, w tym podejściom deliberatywnym, reaktywnym i hybrydowym. Ponadto zbadaliśmy systemy wieloagentowe, w których wiele agentów współpracuje i koordynuje swoje działania, aby osiągnąć wspólne cele poprzez mechanizmy takie jak współpraca, koordynacja i negocjacje.

Wiedza zdobyta w tym rozdziale stanowi solidną podstawę do tworzenia inteligentnych i autonomicznych systemów zdolnych do skutecznego działania w złożonych, nieprzewidywalnych środowiskach. Powinieneś teraz umieć zdecydować, która architektura systemu agentowego najlepiej pasuje do konkretnego przypadku użycia, oraz być w stanie stworzyć mentalny model systemu wieloagentowego, który posłuży jako fundament Twojego systemu agentowego. W następnym rozdziale zagłębimy się w istotę systemów agentowych, co jeszcze bardziej wzmocni Twoje umiejętności budowania wydajnych systemów.

Pytania

1. Jakie są kluczowe cechy inteligentnych agentów?
2. Jakie są główne typy wzorców architektonicznych inteligentnych agentów?
3. Czym różnią się architektury deliberatywne i reaktywne pod względem mocnych i słabych stron?
4. Czym jest system wieloagentowy i jakie są jego główne cechy?
5. Jakie są główne mechanizmy interakcji w systemach wieloagentowych?
6. W jakich dziedzinach najczęściej stosuje się systemy wieloagentowe?

Odpowiedzi

1. Kluczowe cechy inteligentnych agentów to reaktywność, proaktywność, zdolności społeczne, autonomia oraz umiejętność uczenia się i adaptacji.
2. Główne wzorce architektoniczne inteligentnych agentów to architektury deliberatywne (oparte na wiedzy), reaktywne (oparte na zachowaniu) oraz hybrydowe.
3. Architektury deliberatywne doskonale radzą sobie ze złożonym rozumowaniem i planowaniem, ale mogą mieć trudności z szybkim reagowaniem w dynamicznych środowiskach. Architektury reaktywne są dobrze dostosowane do dynamicznych środowisk wymagających szybkich reakcji, lecz brakuje im zdolności długoterminowego planowania i abstrakcyjnego rozumowania.
4. System wieloagentowy składa się z wielu inteligentnych agentów, które współdziałają, współpracują i koordynują swoje działania, aby osiągnąć wspólne cele. Kluczowe cechy takich systemów to autonomia, interakcja, zdolność adaptacji, rozproszona kontrola, skalowalność, heterogeniczność oraz zdecentralizowane dane i wiedza.
5. Główne mechanizmy interakcji w systemach wieloagentowych to współpraca (dążenie do wspólnego celu), koordynacja (zarządzanie współzależnościami) oraz negocjacje (osiąganie porozumień).
6. Systemy wieloagentowe znajdują zastosowanie w takich dziedzinach jak zarządzanie łańcuchem dostaw, kontrola ruchu, robotyka, monitorowanie środowiska, rozproszone sieci czujników oraz inteligentne środowiska wirtualne.

Skorowidz

A

ACL, Agent Communication Language, 172
adaptacja, 101, 189
  dynamiczna, 183
agent AI, *Patrz także* agenty inteligentne
  CrewAI, 107, 110
  oparty na LLM-ie, 35, 100, 132
  przejrzystość samowyjaśniania, 116
  sprawczość i autonomia, 54
  uczenie się i doskonalenie, 117
agentowa sztuczna inteligencja, 92
agenty inteligentne
  algorytmy planowania, 138
    bardziej praktyczne, 141
    mniej praktyczne, 139
  cechy, 57
  cykl pracy
    optymalizacja, 198
    przetwarzanie równoległe, 197
    przetwarzanie sekwencyjne, 196
  historia interakcji, 193
  introspekcja, 104
  mechanizmy uczenia się, 83
  model CWD, 159
    projektowanie, 164–171
  modelowanie środowiska, 186
  narzędzia
    definiowanie, 133
    planowanie wykorzystania narzędzi, 149
    rodzaje, 134
    rozumowanie o narzędziach, 148
    sposoby wykorzystania, 135
    udostępnianie, 130
    użycie AutoGen, 152
    użycie CrewAI, 151
    użycie LangGraph, 154
  określanie celów, 180
  pamięć
    długotrwała, 192
    epizodyczna, 193
    krótkotrwała, 190
  proces podejmowania decyzji, 195
  przestrzenie stanów, 184
  refleksja, 100
    adaptacja, 101
    interakcja człowiek – komputer, 103
    kwestie etyczne, 102
    metapoznawanie, 100
    metarozumowanie, 105
    samomodelowanie, 118
    samowyjaśnianie, 114
  refleksyjne
    czatboty obsługujące klientów, 121
    opracowujące strategie marketingowe, 122
    optymalizujące strategie cenowe, 125
    prognozujące sprzedaż, 124
    w systemach transakcji finansowych, 123
  repozytorium wiedzy, 192
  reprezentacja wiedzy, 74
    oparta na logice, 77
    ramki, 76
    sieci semantyczne, 74
  rozszerzanie możliwości, 91
  rozumowanie
    abdukcyjne, 82
    dedukcyjne, 79
    indukcyjne, 80
  specyfikacje zadań, 181
  świadomość kontekstowa, 183
  wnioskowanie tradycyjne, 105
  wzorce integracji, 188
  zarządzanie kontekstem, 194
AGI, artificial general intelligence, 257, 261
AI, artificial intelligence, 27, 51

algorytmy
planowania, 88, 138
GraphPlan, 140
MCTS, 140
planowanie A*, 139
planowanie FF, 140
planowanie HTN, 144
planowanie oparte na LLM-ie, 142
STRIPS, 139
redukowania stronniczości, 212
API, application programming interface, 38, 93, 134
architektura
pamięci agenta, 190
systemu agentowego, 58
deliberatywna, 60
hybrydowa, 62
reaktywna, 61
transformera, 31
asystent
planowania podróży, 162
rezerwacji lotów, 35
rezerwacji podróży, 54, 69, 92
ataki adwersarialne, 223
autokodery wariacyjne, VAE, 29
Beta-VAE, 29
warunkowe, CVAE, 30
autonomia, autonomy, 53
autonomiczne cykle pracy, 41

B

baza wiedzy, 192
BERT, 33
Beta-VAE, 29
bezpieczeństwo systemów agentowych, 230, 236
bezpieczne
obliczenia wielostronne, SMPC, 235
uczenie się, 233
biblioteka LangChain, 92, 130

C

cofanie zmian, 230, 232
CoT, chain of thought, 38
CVAE, conditional variational autoencoder, 30
CWD, coordinator-worker-delegator, 158, *Patrz także* model CWD

D

DCGAN, Deep Convolutional GAN, 31
dezinformacja, 225
duże modele językowe, LLM, 33
agenty AI, 35, 100, 132
autoregresyjne LLM-y, 33
dostosowane do instrukcji, 34
formatowanie instrukcji, 175
oparte na koderze-dekoderze, 34
oparte tylko na koderze, 34
prompty systemowe, 174
specyficzne dla danej dziedziny, 34
wielomodalne, 34
wywoływanie funkcji i narzędzi, 131

E

etyka, 215, 234

F

FF, Fast Forward, 140
formatowanie instrukcji, 175
framework
AutoGen, 152, 155
CrewAI, 100, 107, 115, 151, 155
LangGraph, 92, 154, 155
funkcja użyteczności, utility function, 85

G

GAN, Generative Adversarial Networks, 30, 258
generatywna sztuczna inteligencja, 27, 49
autokodery wariacyjne, 29
budowanie zaufania, 205
doskonalenie inteligentnych agentów, 91
modele autoregresyjne, 31
narzędzia otwartoźródłowe, 41–43, 45
ograniczenia, 45
jakość danych, 45
kreatywność, 48
kwestie etyczne i społeczne, 48
prywatność danych, 46
uogólnianie, 48
zasoby obliczeniowe, 47
sieć GAN, 30
wdrażanie modelu CWD, 174
zagrożenia, 222
zastosowania, 40–43, 45

generatywne sieci przeciwstawne, GAN, 30, 258
DCGAN, 31
StyleGAN, 31
WGAN, 31
generowanie
kodu, 41
muzyki i dźwięków, 40
obrazów i filmów, 40
tekstu i treści, 40
GPT, Generative Pre-trained Transformer, 33
grafy, 88
granice działania, 230, 231

H

halucynacje, 225
hierarchiczna sieć zadań, HTN, 144
historia interakcji, 193

I

instrukcje dla agentów, 180
inteligencja multimodalna, 259
inteligentny agent, 57 *Patrz* agenty inteligentne
interakcja w systemach wieloagentowych, 66
interfejs programowania aplikacji, API, 38, 93, 134
introspekcja, 104
inżynieria systemów, 51

J

jednokierunkowe sieci neuronowe, 32
język komunikacji agentów, ACL, 172

K

kodowanie pozycyjne, positional encoding, 32
kontekst
efektywne zarządzanie, 194
globalny, 194
sesji, 194
świadomość kontekstowa, 183
zadania, 194
kontrola dostępu oparta na rolach, RBAC, 230
koordynacja w systemach wieloagentowych, 67
koordynator – wykonawca – delegator, CWD, 158, *Patrz także* model CWD

L

LLM, large language model, 33
logika, 77

Ł

łańcuch myśli, CoT, 38

M

małe modele językowe, SLM, 46
MAML, Model-Agnostic Meta-Learning, 263
mapa istotności, 208, 210
MAS, multi-agent system, 63
MCTS, 140
mechanizm samouwagi, self-attention mechanism, 31
metapoznanie, metacognition, 100
metarozumowanie, meta-reasoning, 105, 110
metauczenie, 263
metoda Monte Carlo, 89
model CWD, 159, 178
delegator, 160, 162
dzielenie się wiedzą, 173
implementacja, 161
komunikacja agentów, 172
koordynator, 159, 161
negocjacje, 172
projektowanie agentów, 164–171
rola
Analityk, Analyst, 165
Interpretator zadań, Task interpreter, 166
Menedżer, Manager, 165
Poszukiwacz, Searcher, 165
Ulepszacz, Reflector, 165
rozwiązywanie konfliktów, 172
w systemach generatywnej AI, 174
współpraca agentów, 172
wykonawca, 160, 162
wzorce interakcji między agentami, 176
zasady, 160
modele
autoregresyjne, 31
generatywnej sztucznej inteligencji, 29
modelowanie środowiska, 186
monitorowanie, 189
w czasie rzeczywistym, 231, 232
wydajności, 111

N

naruszenia prywatności, 227
narzędzia
  API, 134
  bazodanowe, 134
  definiowanie
    z użyciem docstringów, 133
    z użyciem schematu JSON, 133
  funkcje użytkowe, 135
  integracyjne, 135
  interfejsu sprzętowego, 135
  w systemach agentowych, 137
  wykorzystujące generatywną AI, 41–43, 45
negocjacje w systemach wieloagentowych, 68
NER, named entity recognition, 34
niepewność, 211, 217
NLP, natural language processing, 31
NLU, natural language understanding, 34
normalizacja warstw, layer normalization, 32

O

ochrona danych, 235
odpowiedzialność za wyniki, 234
ogólna sztuczna inteligencja, AGI, 257, 261
opracowywanie leków, 40
optymalizacja cyklu pracy, 198

P

pamięć
  długotrwała, 192
  epizodyczna, 193
  krótkotrwała, 190
PixelCNN, 33
PixelSNAIL, 33
planowanie
  algorytmy, 88, 138–142, 144
  hierarchiczne, 90
  oparte na grafach, 88
połączenia resztkowe, residual connections, 32
problemy spełnialności więzów, CSP, 90
projektowanie
  agentów, 164–171
  systemów agentowych, 179
    architektura pamięci, 190
    cykle pracy, 196
    modelowanie środowiska, 186
    prompty systemowe, 180
    przestrzenie stanów, 184
prompty systemowe, 174
  ukierunkowane, 180
prywatność, 235, 236
  danych, 227
przejrzystość, 207, 216
przestrzenie stanów, 184
przeszukiwanie drzew, 89
  metodą Monte Carlo, MCTS, 140
przetwarzanie
  języka naturalnego, NLP, 31, 244
  równoległe, 197
  sekwencyjne, 196
Python
  implementacja agenta, 56

R

ramki, frame, 76
refleksja w agentach, 100–114
reprezentacja przestrzeni stanów, 185
robotyka, 41, 51, 247, 255
role, 164
rozplątywanie cech, 30
rozpoznawanie jednostek nazewniczych, NER, 34
rozumienie
  języka naturalnego, NLU, 34
  zaawansowane języka, 259
rozumowanie
  abdukcyjne, 82
  dedukcyjne, 79
  indukcyjne, 80

S

samomodelowanie, self-modeling, 118
samorządność, self-governance, 51
samouwaga, 31
samowyjaśnianie, self-explananation, 114
sieci semantyczne, 74
sieć GAN, 30
  głęboka splotowa, DCGAN, 31
SLM, small language model, 46
spełnialność więzów, 90
sprawczość, agency, 52
sterownik agenta, agent controller, 131
STRIPS, Stanford Research Institute Problem Solver, 139
StyleGAN, 31

systemy
agentowe, agentic system, 50, 258
agenty artystyczne, 241
agenty inteligentne, 57
agenty językowe, 244, 245
agenty kreatywne, 241
agenty robotyczne, 247
architektury deliberatywne, 59
architektury hybrydowe, 62
architektury reaktywne, 60
autonomia, 53
bezpieczeństwo, 230
funkcja użyteczności, 85
inżynieria systemów, 51
planowanie, 88
podejmowanie decyzji, 85
robotyka, 51, 247, 255
samorządność, 51
sprawczość, 52
systemy autonomiczne, 247
sztuczna inteligencja, 51
techniki projektowania, 92, 179
wspomaganie decyzji, 250, 251
wykorzystywanie narzędzi, 137
zastosowania kreatywne, 241
wieloagentowe, MAS, 63, 166
cechy, 65
definicja, 65
dzielenie się wiedzą, 173
komunikacja agentów, 172
mechanizm koordynacji, 172
mechanizmy interakcji, 66
negocjacje, 68, 172
oparte na modelu CWD, 159, 171
projektowanie agentów, 164–171
rozwiązywanie konfliktów, 172
sztuczna inteligencja, AI, 27, 51

Ś

świadomość kontekstowa, 183

T

T5, Text-To-Text Transfer Transformer, 33
transformer
struktura modelu, 32
trening przeciwstawny, adversarial training, 212
tworzenie systemów agentowych, 92, 179

U

uczenie maszynowe, 27
chroniące prywatność, PPML, 235
nadzorowane, supervised learning, 84
nienadzorowane, unsupervised learning, 84
przez wzmacnianie, reinforcement learning, 84, 114, 260
przez wzmacnianie z informacją zwrotną, RLHF, 231
samonadzorowane, self-supervised learning, 262
się na podstawie kilku przykładów, 259, 264
się przez doświadczenie, 260
transferowe, transfer learning, 84, 263
z udziałem człowieka, HITL, 234
uprzedzenia, 211, 217
uwaga wielowątkowa, multi-head attention, 32

V

VAE, Variational Autoencoders, 29

W

wąska sztuczna inteligencja, narrow AI, 257
weryfikacja decyzji, 230, 232
WGAN, Wasserstein GAN, 31
wizualizacja uwagi, 208, 209
własność intelektualna, 228
wnioskowanie
przyczynowe, causal reasoning, 262
tradycyjne, traditional reasoning, 105
wskaźniki wydajności, 111, 189, 231, 232
wyjaśnialna sztuczna inteligencja, XAI, 208
wyjaśnialność, 207, 216
wyjaśnienia w języku naturalnym, 208, 211
wyszukiwanie heurystyczne, 89
wywoływanie
funkcji, function calling, 131
narzędzi, tool calling, 131
wzorce
integracji, 188
interakcji, 176, 188

X

XAI, explainable AI, 208

Z

zagrożenia generatywnej AI, 222
  ataki adwersarialne, 223
  dezinformacja, 225
  halucynacje, 225
  naruszenia prywatności danych, 227
  uprzedzenia i dyskryminacja, 224
  związane z własnością intelektualną, 228
zarządzanie kontekstem, 194
zastosowania systemów agentowych
  agenty konwersacyjne, 244
  optymalizacja, 250
  przetwarzanie języka naturalnego, 244
  robotyka, 247
  systemy autonomiczne, 247
  w różnych branżach, 260
  wspomaganie decyzji, 250
  zastosowania kreatywne i artystyczne, 241
zaufanie do generatywnej AI, 205, 206
  etyka i odpowiedzialność, 215
  kontrola i zgoda użytkownika, 214
  mapa istotności, 210
  ograniczanie niepewności i uprzedzeń, 211, 217
  przejrzystość i wyjaśnialność, 207, 216
  skuteczne komunikowanie wyników, 213
  wizualizacja uwagi, 209
  wyjaśnienia w języku naturalnym, 211
zgody użytkownika, 214

Ta książka to nie tylko kompendium techniczne. Zachęca badaczy, liderów branży i innowatorów do aktywnego udziału w tworzeniu kolejnego rozdziału AI!

— Matthew R. Scott, CTO, Minset.ai

Generatywna AI i systemy agentowe w niewyobrażalny wcześniej sposób transformują liczne branże, umożliwiając maszynom generowanie tekstu, obrazów, a nawet planów strategicznych dzięki zdolności do analizy kontekstu i adaptacji do zmieniających się warunków. Odpowiedzialne zastosowanie agentów AI wymaga jednak od architektów głębokiego zrozumienia zasad ich działania.

Dzięki tej książce poznasz podstawy generatywnej sztucznej inteligencji i architektury agentowej, a także nauczysz się projektować systemy zdolne do planowania, uczenia się i samodoskonalenia. Autorzy opisują kluczowe struktury decyzyjne, mechanizmy adaptacji i techniki planowania wieloetapowego. Znajdziesz tu omówienie integracji z narzędziami zewnętrznymi, strategii koordynatora–wykonawcy–delegatora w skalowalnych agentach AI, jak również zagadnienia związane z bezpieczeństwem, zaufaniem i etyką — niezbędne do tworzenia systemów zgodnych z ludzkimi wartościami. Zapoznasz się ponadto z rzeczywistymi zastosowaniami agentowej sztucznej inteligencji w automatyce, finansach czy ochronie zdrowia.

Najważniejsze zagadnienia:

- podstawy generatywnej AI i systemów agentowych
- agenty AI w dynamicznych środowiskach
- analizowanie własnych działań agenta AI i improwizowanie
- planowanie złożonych zadań i korzystanie z zewnętrznych narzędzi
- zwiększanie przejrzystości, odpowiedzialności i niezawodności AI
- zastosowanie agentów AI w różnych branżach

Anjanava Biswas specjalizuje się w generatywnej sztucznej inteligencji, przetwarzaniu języka naturalnego, uczeniu głębokim, analizie danych i architekturze chmurowej. Publikuje prace badawcze w wielu czasopismach naukowych.

Wrick Talukdar pełni funkcję lidera technologicznego w dziedzinie generatywnej sztucznej inteligencji w Amazonie. Jest autorem wielokrotnie cytowanych prac badawczych w obszarze GenAI, NLP i widzenia komputerowego.

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 978-83-289-3319-4

Cena: 89,00 zł

packt