МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ЛЕКЦІЯ 12. Системи AI агентів: архітектура, протокол контексту моделі та мультиагентні системи

Львів -- 2025

Лекція курсу "Штучний інтелект в ігрових застосунках" 2025-12

Вступ

У цій лекції ми розглянемо концепцію AI агентів, їх архітектуру та застосування. Ми сфокусуємось на протоколі контексту моделі (Model Context Protocol), що дозволяє агентам ефективно взаємодіяти з даними та інструментами, а також на мультиагентних системах, де декілька агентів співпрацюють для вирішення складних завдань.

Агенти на базі великих мовних моделей (LLM) стають все більш поширеними інструментами для автоматизації складних робочих процесів. Вони можуть виконувати різноманітні завдання, від написання коду до взаємодії з користувачами в режимі реального часу. Розуміння того, як проектувати, розробляти та використовувати такі системи, є ключовим для майбутніх фахівців з штучного інтелекту.

Ця лекція базується на останніх досягненнях у галузі агентних систем. Ми також розглянемо етичні аспекти та майбутні напрямки розвитку цієї технології.

Що таке AI агенти?

AI агент — це програмна система, яка використовує штучний інтелект для досягнення цілей та виконання завдань від імені користувачів. Агенти мають певний рівень автономності для прийняття рішень, навчання та адаптації.

Коли ми говоримо про сучасних AI агентів, ми зазвичай маємо на увазі системи, які базуються на великих мовних моделях (LLM) і можуть виконувати складні робочі процеси автоматично. Наприклад, агент може допомогти користувачеві зробити онлайн-замовлення товарів, порівняти різні варіанти, і організувати їх доставку в маркетплейсі.

Ключові характеристики AI агентів

Ефективний AI агент має ряд важливих характеристик:

- **Міркування (Reasoning)**: Процес доповнення входного вектора даних проміжковими результатами генерації, які симулюють процес міркування. Є різновидністю промпт-інжинерії і дозволяє поліпшувати якість фінальної генерації мовної моделі, як і інші методики промпт-інжинерії, наприклад, методика запитів з ілюстраціями (few-shot prompting).
- **Інструменти (Tooling)**: Здатність виконувати дії або завдання на основі рішень, планів або зовнішніх вхідних даних є вирішальною для AI агентів, щоб взаємодіяти з їхнім середовищем і досягати цілей. Це може включати фізичні дії у випадку втіленого AI, або цифрові дії, такі як надсилання повідомлень, оновлення даних або запуск інших процесів.
- **Організація (Orchestration)**: Організація дій системагентів у часі і просторі.

- **Довострокова пам'ять (Long-term memory)**: Здатність використовувати інструменти для зберігання інформації на тривалий термін, щоб використовувати її для збагачення промптів і, таким чином, кращого прийняття рішень і виконання завдань.
- **Структуровані результати (Structured outputs)**: Здатність виводити результати у вигляді структурованих даних, таких як таблиці, списки або JSON з метаданими про походження результату. Структуровані результати дозволяють організовувати роботу груп різнорідних агентів і є базою для мультиагентних систем і створення стандартів обміну інформацією між агентами.
- **Стандартизований обмін інформацією (Standardized information exchange)**: Стандартизований обмін інформацією між агентами дозволяє автоматизувати створення адаптерів для інструментів, тобто надати можливість доступу до широкого спектра інструментів і моделей.

Базова структура AI агента

Ось приклад базової структури AI агента, побудованого на базі бібліотеки Agno, який використовує модель Claude 3.7 Sonnet і виконує завдання написання звіту про компанію NVDA:

```
from agno.agent import Agent
from agno.models.anthropic import Claude
from agno.tools.reasoning import ReasoningTools
from agno.tools.yfinance import YFinanceTools

agent = Agent(
    model=Claude(id="claude-3-7-sonnet-latest"),
    tools=[
        ReasoningTools(add_instructions=True),
        YFinanceTools(stock_price=True, analyst_recommendations=True, company_info=True, compar
    ],
    instructions=[
        "Use tables to display data",
        "Only output the report, no other text",
    ],
    markdown=True,
)
agent.print_response("Write a report on NVDA", stream=True, show_full_reasoning=True, stream_ir
```

У цьому прикладі ми визначаємо:

- Модель ШІ, яку використовуватиме агент (у даному випадку, Claude 3.7 Sonnet)
- Інструменти для міркування та фінансових даних
- Інструкції для форматування звіту
- Параметри для виведення результату у вигляді таблиці

Цей код створює ефективного AI агента, який може виконувати складні завдання, такі як написання звіту про компанію, використовуючи здатність LLM і інструменти для аналізу даних.

Відмінності між AI агентами, асистентами та ботами

Важливо розрізняти AI агентів, асистентів та ботів:

AI агенти автономно та проактивно виконують завдання, можуть виконувати складні, багатоетапні дії, навчаються та адаптуються, можуть приймати рішення самостійно. Вони проактивні та орієнтовані на цілі.

AI асистенти допомагають користувачам із завданнями, відповідають на запити або підказки, надають інформацію та виконують прості завдання, можуть рекомендувати дії, але користувач приймає рішення. Вони реактивні, реагуючи на запити користувачів.

Боти автоматизують прості завдання або розмови, слідують заздалегідь визначеним правилам, мають обмежені можливості навчання, підтримують базові взаємодії. Вони реактивні, реагуючи на тригери або команди.

Ключові відмінності:

- **Автономність**: AI агенти мають найвищий ступінь автономності, AI асистенти менш автономні, а боти найменш автономні.
- **Складність**: AI агенти призначені для обробки складних завдань і робочих процесів, тоді як AI асистенти та боти більше підходять для простіших завдань і взаємодій.
- **Навчання**: AI агенти часто використовують машинне навчання для адаптації та покращення своєї продуктивності з часом. AI асистенти можуть мати деякі можливості навчання, тоді як боти зазвичай мають обмежене навчання або не мають його взагалі.

Протокол контексту моделі (MCP)

Model Context Protocol (MCP) — це відкритий протокол, який стандартизує спосіб надання контексту для LLM (великих мовних моделей). MCP забезпечує стандартизований спосіб підключення AI-моделей до різних джерел даних та інструментів.

Основна архітектура MCP

MCP реалізує клієнт-серверну архітектуру, де хост-додаток може підключатися до декількох серверів:

- **MCP Хости (Hosts)**: Програми, такі як Claude Desktop, IDE або AI-інструменти, які хочуть отримати доступ до даних через MCP
- **MCP Клієнти (Clients)**: Клієнти протоколу, які підтримують з'єднання 1:1 із серверами
- **MCP Сервери (Servers)**: Легкі програми, кожна з яких надає певні можливості через стандартизований протокол MCP
- **Локальні джерела даних (Local Data Sources)**: Файли комп'ютера, бази даних та служби, до яких сервери MCP можуть безпечно отримати доступ
- **Віддалені сервіси (Remote Services)**: Зовнішні системи, доступні через інтернет (наприклад, через API), до яких можуть підключатися сервери MCP

Основні компоненти MCP

MCP визначає кілька ключових компонентів, які дозволяють моделям ШІ взаємодіяти з зовнішніми джерелами даних і інструментами:

1. **Ресурси (Resources)**: Дозволяють серверам надавати доступ до контексту та даних як для користувача, так і для AI-моделі.
2. **Підказки (Prompts)**: Шаблонні повідомлення та робочі процеси для користувачів.
3. **Інструменти (Tools)**: Функції, які AI-модель може виконувати.
4. **Семплінг (Sampling)**: Функціональність, що дозволяє серверам ініціювати агентну поведінку та рекурсивні LLM-взаємодії.

Безпека та конфіденційність в MCP

MCP надає потужні можливості через довільний доступ до даних та шляхи виконання коду. З цією міццю пов'язані важливі міркування щодо безпеки та довіри, які всі реалізатори повинні ретельно враховувати.

Ключові принципи:

1. **Згода та контроль користувача**
    - Користувачі повинні явно погоджуватися та розуміти весь доступ до даних та операції
    - Користувачі повинні зберігати контроль над тим, які дані передаються і які дії виконуються
    - Реалізатори повинні забезпечити чіткі інтерфейси для перегляду та авторизації діяльності
2. **Конфіденційність даних**
    - Хости повинні отримати явну згоду користувача перед наданням даних користувача серверам
    - Хости не повинні передавати дані ресурсів деінде без згоди користувача
    - Дані користувача повинні бути захищені відповідними контролями доступу
3. **Безпека інструментів**
    - Інструменти представляють довільне виконання коду і до них слід ставитися з відповідною обережністю
    - Хости повинні отримати явну згоду користувача перед викликом будь-якого інструменту
    - Користувачі повинні розуміти, що робить кожен інструмент, перш ніж дозволяти його використання

Як MCP підтримує AI агентів

MCP особливо корисний при побудові AI агентів, оскільки він забезпечує:

1. **Доступ до даних**: Агенти можуть безпечно отримувати доступ до локальних файлів, баз даних та інших джерел інформації.
2. **Стандартизовані інструменти**: Агенти можуть використовувати інструменти для взаємодії з зовнішніми системами.
3. **Переносимість між моделями**: Реалізації MCP працюють з різними провайдерами LLM.
4. **Безпечні рекурсивні виклики**: Агенти можуть викликати інші моделі для виконання піддій.

MCP підтримує стандартний формат JSON-RPC для обміну повідомленнями між клієнтами та серверами, що полегшує інтеграцію з різними системами. З точки зору розробників, MCP реалізований через різні SDK для мов програмування, включаючи Python, TypeScript, Java, Kotlin та C#.

Приклад використання MCP

Розглянемо простий приклад, як агент може використовувати MCP для отримання доступу до локальних файлів:

```
from mcp_python.client import MCPClient
from mcp_python.resources import ResourcesClient

# Створення клієнта MCP
client = MCPClient("localhost:8080")

# Отримання доступу до ресурсів
resources_client = ResourcesClient(client)

# Читання файлу через MCP
file_content = resources_client.get_resource("/path/to/file.txt").decode("utf-8")

# Використання вмісту файлу в агенті
agent.prompt(f"Проаналізуй цей текст: {file_content}")
```

У цьому прикладі агент використовує MCP для читання файлу з локальної файлової системи. Це дозволяє агенту отримати доступ до даних, необхідних для виконання завдання, при цьому зберігаючи модель і дані розділеними, що покращує безпеку та гнучкість.

Мультиагентні системи

Мультиагентна система (MAS) — це середовище, де кілька незалежних агентів працюють разом для вирішення складних завдань. Ці агенти можуть співпрацювати, координуватися або навіть конкурувати, залежно від цілей системи.

Відмінність від одноагентних систем

Одноагентні системи покладаються на одного агента, який послідовно виконує всі задачі. Хоча такий підхід працює для простих ситуацій, він має суттєві обмеження при зростанні складності:

1. **Обмежена спеціалізація**: Один агент повинен бути "майстром на всі руки".
2. **Відсутність паралелізму**: Завдання виконуються послідовно, що обмежує продуктивність.
3. **Єдина точка відмови**: Якщо агент виходить з ладу, вся система припиняє роботу.
4. **Обмежене масштабування**: Одному агенту складно обробляти складні завдання з багатьма підзадачами.

Мультиагентні системи вирішують ці проблеми, розподіляючи роботу між кількома спеціалізованими агентами, що підвищує ефективність, стійкість та здатність до масштабування.

Основні компоненти мультиагентних систем

Ефективна мультиагентна система складається з таких ключових компонентів:

1. **Агенти**: Автономні сутності, здатні сприймати середовище, приймати рішення та виконувати дії для досягнення конкретних цілей.

2. **Середовище**: Світ, у якому працюють агенти. Може бути фізичним (наприклад, робототехнічна майстерня) або віртуальним (наприклад, симульований ринок). Середовище надає контекст для дій та взаємодій агентів.

3. **Організаційна структура**: Визначає, як агенти розташовані та пов'язані один з одним у системі. Структура може бути ієрархічною, командною або повністю децентралізованою, залежно від цілей та філософії дизайну системи.

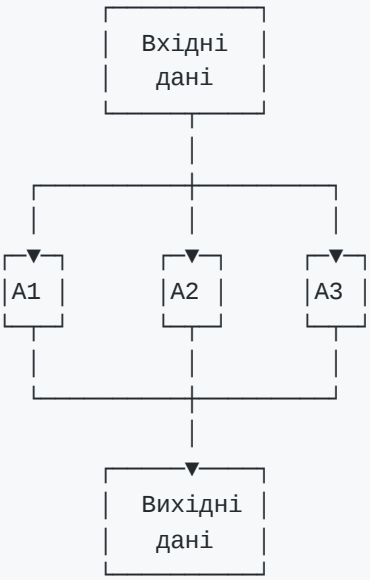
4. **Взаємодії**: Взаємодії між агентами мають вирішальне значення для функціональності системи. Вони можуть включати протоколи зв'язку, механізми переговорів та стратегії координації. Ефективні взаємодії дозволяють агентам обмінюватися інформацією, вирішувати конфлікти та працювати разом для досягнення спільних цілей.

Патерни архітектур мультиагентних систем

В мультиагентних системах існує кілька поширених патернів організації агентів:

1. Паралельний патерн

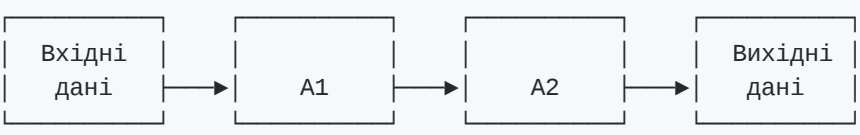
Агенти працюють незалежно і паралельно над різними задачами, що дозволяє системі обробляти кілька завдань одночасно.



Приклад: система аналізу ринку, де різні агенти одночасно аналізують різні сегменти ринку.

2. Послідовний патерн

Агенти працюють послідовно, де вихід одного агента стає входом для іншого.



Приклад: конвеєр обробки даних, де перший агент збирає дані, другий їх очищає, а третій аналізує.

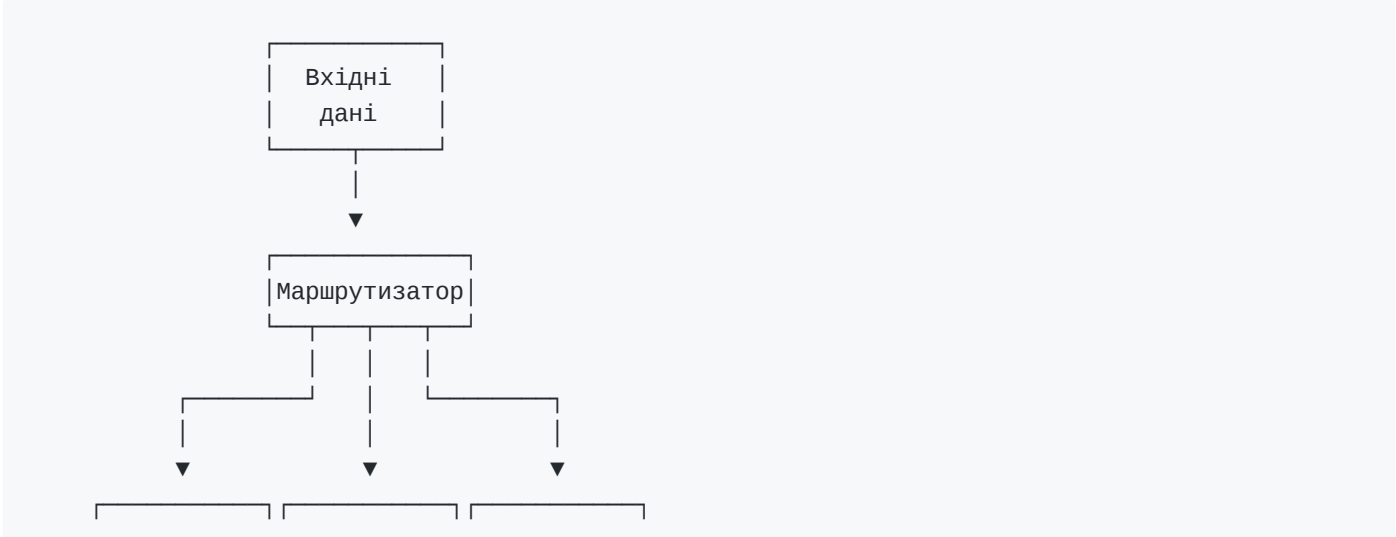
3. Цикл

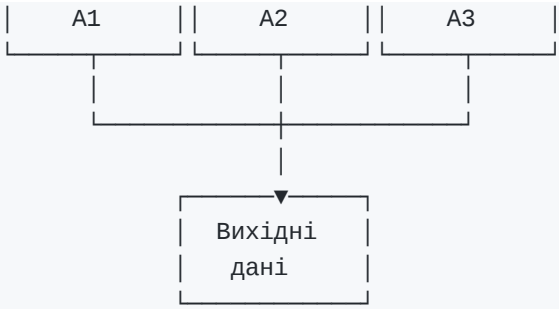
Агенти працюють в циклі, де результати можуть повертатися до попередніх агентів для уточнення.

Приклад: система творчого письма, де один агент генерує текст, інший критикує його, а третій вносить поліпшення, після чого цикл може повторюватися.

4. Маршрутизатор

Агент-маршрутизатор перенаправляє завдання найбільш відповідному агенту на основі типу завдання.

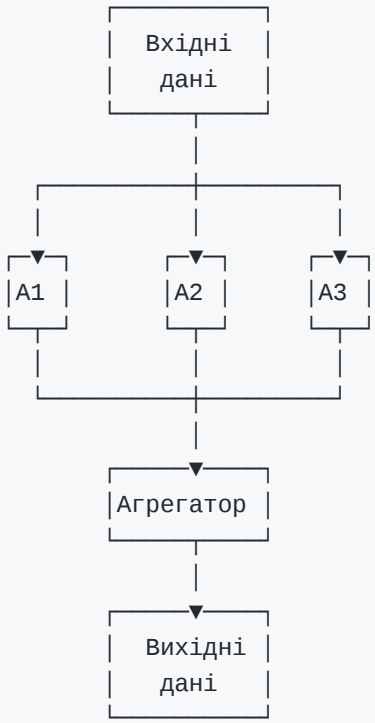




Приклад: система підтримки клієнтів, де маршрутизатор направляє запити клієнтів до агентів, що спеціалізуються на конкретних типах проблем.

5. Агрегатор (або Синтезатор)

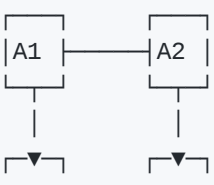
Кілька агентів працюють над задачею, а потім агрегатор об'єднує їхні результати.

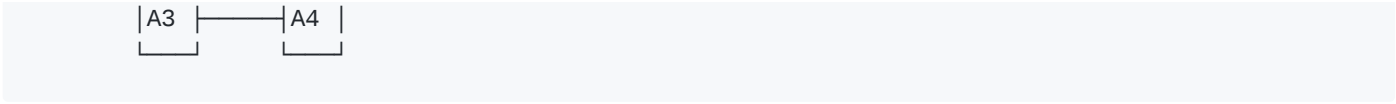


Приклад: система аналізу ризиків, де різні агенти оцінюють різні аспекти ризику, а агрегатор об'єднує ці оцінки в загальний профіль ризику.

6. Мережевий (або Горизонтальний)

Агенти організовані в мережу та можуть вільно спілкуватися один з одним без централізованого контролю.

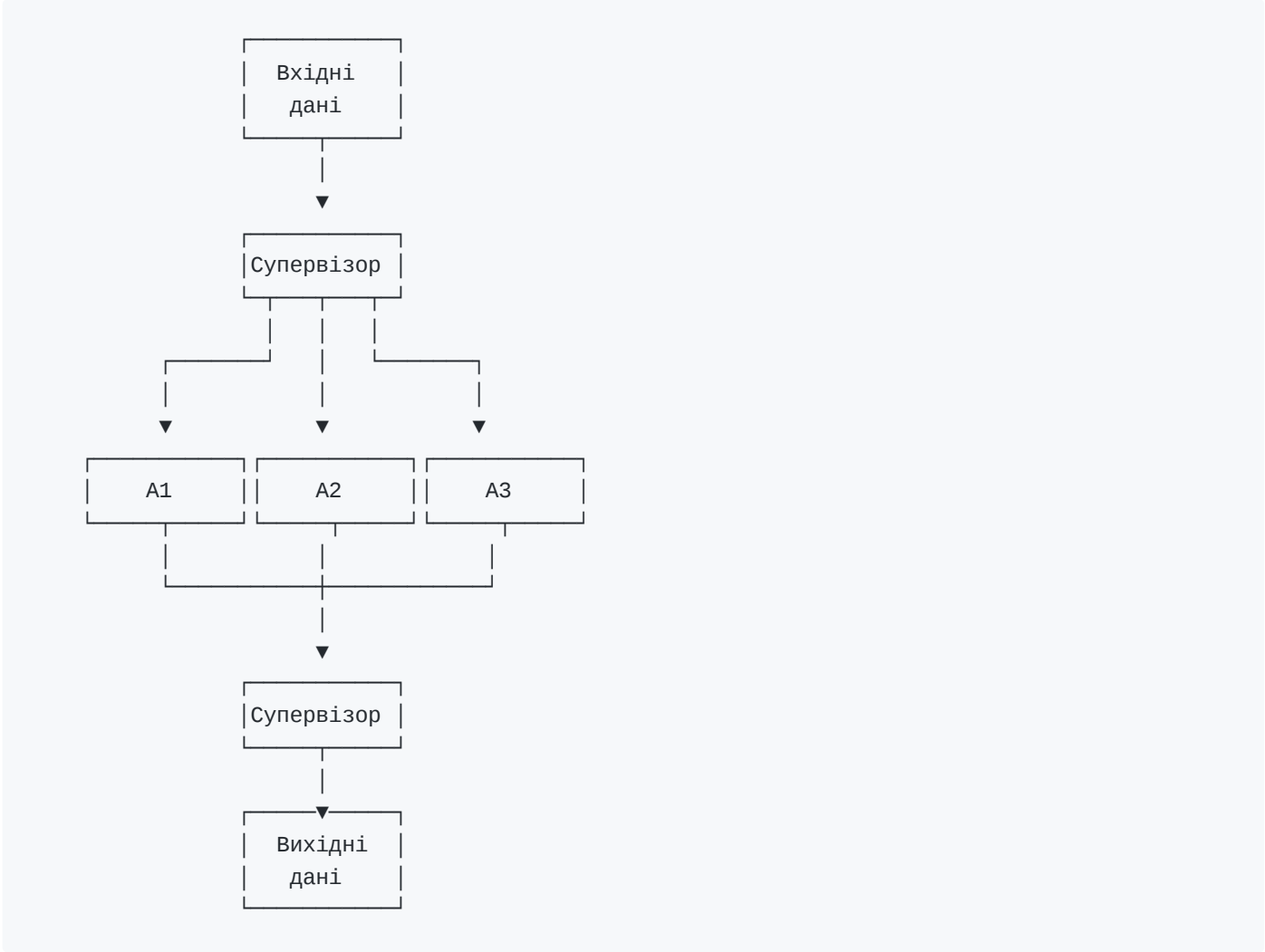




Приклад: система моделювання ринку, де агенти представляють окремих торговців, які взаємодіють, щоб встановити ціни та виконати угоди.

7. Супервізор

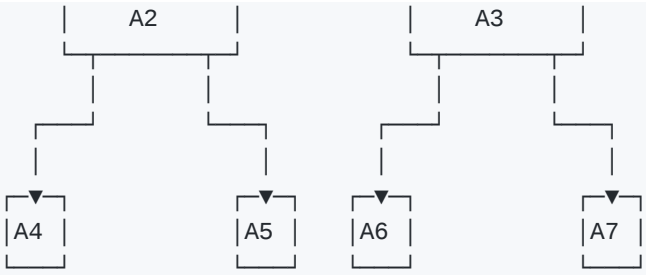
Супервізор координує роботу інших агентів, встановлюючи цілі та слідкуючи за виконанням.



Приклад: проектна команда, де агент-менеджер призначає завдання, встановлює терміни та відстежує прогрес інших агентів.

8. Ієрархічний (або Вертикальний)

Агенти організовані в ієрархію, де агенти на вищих рівнях контролюють агентів на нижчих рівнях.



Приклад: корпоративна структура, де виконавчі агенти приймають стратегічні рішення, менеджери середньої ланки перетворюють їх у тактичні плани, а агенти на нижчому рівні виконують конкретні завдання.

Комунікація між агентами

Для ефективної роботи мультиагентної системи критично важливо налагодити комунікацію між агентами. Існує кілька поширених підходів:

Стан графа vs виклики інструментів

Агенти можуть спілкуватися через спільний стан графа, де інформація доступна всім агентам, або через виклики інструментів, де один агент викликає функцію іншого агента.

```
# Спілкування через стан графа
def agent1(state):
    # Прочитати дані зі стану
    data = state.get("data")
    # Обробити дані
    processed_data = process(data)
    # Оновити стан
    state.set("processed_data", processed_data)
    return state

# Спілкування через виклики інструментів
def agent2(query):
    # Агент 2 має доступ до функціональності агента 1 як інструменту
    tool = get_tool("agent1_processor")
    result = tool.call(query=query)
    return analyze(result)
```

Різні схеми стану

Агенти можуть використовувати різні схеми стану для структурування інформації, якою вони обмінюються:

```
# Агент, що працює з файлами
file_agent_state = {
    "files": [{"name": "document.txt", "content": "..."}],
    "actions": ["read", "write", "delete"]
}

# Агент, що аналізує дані
```

```
data_agent_state = {
    "data": {"type": "time_series", "values": [1, 2, 3, 4]},
    "metrics": ["mean", "variance", "trend"]
}
```

Спільний список повідомлень

Найпростіший спосіб комунікації — використання спільного списку повідомлень, доступного всім агентам:

```
messages = []

def agent1():
    messages.append({"role": "agent1", "content": "Ось деякі дані для аналізу..."})

def agent2():
    # Читає останнє повідомлення від agent1
    last_message = [m for m in messages if m["role"] == "agent1"][-1]
    # Відповідає на повідомлення
    messages.append({"role": "agent2", "content": "Я проаналізував дані..."})
```

Майбутнє агентних систем

Технологія агентних систем стрімко розвивається, і ми бачимо кілька важливих тенденцій, які визначатимуть майбутнє цієї галузі.

Технологічні тренди

1. **Посилення автономності**: Майбутні агенти стануть ще більш автономними, здатними діяти з мінімальним наглядом людини. Вони зможуть самостійно приймати рішення, вчитися на досвіді та адаптуватися до мінливих умов.

2. **Покращення міжагентної комунікації**: Протоколи, такі як Model Context Protocol (MCP) та Agent2Agent, будуть вдосконалюватися, забезпечуючи більш ефективну та безпечну комунікацію між агентами, незалежно від того, які фреймворки або вендори їх створили.

3. **Інтеграція з фізичним світом**: Через інтеграцію з IoT (Інтернетом речей), роботами та іншими фізичними системами, агенти зможуть взаємодіяти з реальним світом більш безпосередньо, відкриваючи нові можливості для автоматизації.

4. **Підвищення мультимодальності**: Агенти зможуть працювати з різноманітними типами даних, включаючи текст, зображення, аудіо та відео, що дозволить їм розуміти світ більш комплексно.

Нові застосування

Розвиток агентних технологій відкриває нові сфери застосування:

1. **Персоналізовані життєві асистенти**: Індивідуальні агенти, які знають ваші уподобання, звички та цілі, зможуть допомагати з повсякденними завданнями, створювати персоналізовані рекомендації та оптимізувати ваше життя відповідно до ваших пріоритетів.

2. **Автономні бізнес-операції**: Підприємства зможуть автоматизувати складні бізнес-процеси, які раніше вимагали людського втручання, за допомогою координованих систем агентів.

3. **Наукові дослідження**: Агенти можуть допомагати вченим аналізувати дані, генерувати та тестувати гіпотези, та навіть розробляти експерименти, значно прискорюючи науковий прогрес.

4. **Розширена освіта**: Персоналізовані освітні агенти зможуть адаптувати навчальний процес під кожного студента, надаючи індивідуалізований зворотний зв'язок та підтримку.

Етичні та соціальні виклики

Поширення агентних систем породжує важливі виклики:

1. **Відповідальність та прозорість**: Хто несе відповідальність за дії, виконані автономними агентами? Як забезпечити прозорість прийняття рішень?

2. **Приватність та безпека**: Агенти мають доступ до чутливої інформації. Як захистити дані користувачів?

3. **Соціальні зміни**: Як широке застосування агентів вплине на ринок праці, соціальні взаємодії та соціальні структури?

4. **Регулювання**: Які правові рамки потрібні для регулювання розробки та використання агентних систем?

Вирішення цих викликів вимагатиме співпраці між технологами, політиками, етиками та громадянським суспільством.

Висновки

AI агенти представляють собою потужну парадигму в галузі штучного інтелекту, яка відкриває нові можливості для автоматизації складних процесів, підвищення продуктивності та створення інноваційних рішень. У цій лекції ми розглянули основні концепції, архітектури та практичні аспекти агентних систем.

Ми ознайомилися з протоколом контексту моделі (MCP), який стандартизує взаємодію між моделями ШІ та зовнішніми даними і інструментами, забезпечуючи більшу гнучкість, безпеку та інтероперабельність агентних систем.

Особливу увагу ми приділили мультиагентним системам, які дозволяють кільком спеціалізованим агентам співпрацювати для вирішення складних завдань. Ми розглянули різні патерни архітектури мультиагентних систем та механізми комунікації між агентами.

Ми також обговорили практичні аспекти розробки агентів, популярні фреймворки та кращі практики, а також розглянули приклади застосування агентних систем у різних галузях підприємницької діяльності.

У міру розвитку технологій великих мовних моделей та відповідних інфраструктур, агентні системи стануть ще потужнішими, автономнішими та здатними до більш складної співпраці. Однак, разом з цими можливостями приходять і серйозні етичні та соціальні виклики, які потребують уважного розгляду.

Майбутнє агентних систем залежить не лише від технологічних інновацій, але й від нашої здатності розробляти та впроваджувати ці технології відповідально, з урахуванням їх впливу на суспільство. Як фахівці в галузі штучного інтелекту, ми маємо унікальну можливість формувати це майбутнє, забезпечуючи, щоб агентні системи працювали на благо людства.

Література та додаткові ресурси

1. Google Cloud. (2024). What are AI agents?
2. Model Context Protocol. (2025). Introduction to MCP
3. PromptingGuide.AI. (2025). Few-shot prompting
4. Wikipedia. (2025). Intelligent agent
5. Wikipedia. (2025). Multi-agent system
6. LangChain Guide. (2025). Multi-Agent Systems
7. Awesome MCP Servers
8. Awesome MCP Clients
9. Ksenia Se. (2025). What Is MCP, and Why Is Everyone Talking About It?
10. Agno. (2025). Документація Agno
11. OpenAI. (2024). OpenAI Agents SDK
12. LangChain. (2025). LangGraph Documentation
13. Microsoft. (2025). Autogen Documentation