УДК 004.89, 004.912

В. Шимкович, В. Чимшир, К. Знова,
Д. Ювженко, Г. Новаковський, С. Теленик

СИНТЕЗ АРХІТЕКТУРИ СЕРВІСІВ НА ОСНОВІ НЕЙРОМЕРЕЖІ З ГЛИБОКИМ НАВЧАННЯМ

Анотація: Розглянута проблема синтезу архітектури сервісів, які надаються провайдерам інфокомунікаційних послуг в інформаційних системах ІТ-компаній-розробників за моделлю End-to-End (E2E). Проаналізовано вимоги до архітектур сервісів, розподілено їх на категорії та блоки, також проаналізовано можливі архітектури сервісів. Розроблено математичні моделі для опису вимог до сервісів та їх архітектури. Проаналізовано можливі архітектури нейроних мереж, що можуть бути використані для генерації математичної моделі архітектури сервісу, в результаті аналізу обрано архітектуру трансформер. Проведено навчання моделі трансформер для генерації архітектор сервісів за вимогами до сервісу. Запропонована нейромережева модель та технологія синтезу архітектури сервісів на її основі є складовою платформи підтримки життєвого циклу сервісів у інформаційних системах провайдерів. У інтегральній системі засобів автоматизації процесів життєвого циклу сервісів запропонована нейронна мережа реалізує функціонал для побудови архітектури сервісів.

Ключові слова: проектування сервісів, архітектура сервісу, штучний інтелект, нейронні мережі, глибоке навчання, трансформери.

Вступ

Сервісний підхід зміцнює позиції у ІТ-галузі. Вибір інструментів для його впровадження у різних галузях досить широкий і постійно зростає [1]. Особливо швидко змінюються інструменти підтримки життєвого циклу сервісів в інформаційних системах провайдерів інфокомунікаційних послуг [2]. Цьому сприяє сучасний підхід до створення інформаційних систем провайдерів інфокомунікаційних послуг на основі моделі E2E [3, 4]. Проблема побудови архітектури сервісів в інформаційних системах провайдерів телекомунікацій постала при впровадженні концепції архітектури в сучасних методологіях створення інформаційних систем і їх програмного забезпечення [5-7]. Проектування сервісів в інформаційних системах провайдерів інфокомунікаційних послуг почали розглядати як побудову їх архітектури. Однак відбулася не лише заміна термінів, побудова архітектури сервісів доповнилася новими аспектами і перетворилася в складну проблему, ефективного вирішення якої чекають провайдери. Для вирішення даної проблеми найчастіше зустрічаються технології, які дозволяють подавати архітектуру сервісів за допомогою графічних мов з використанням напрацьованих шаблонів. Загалом ці рішення

є специфічними для кожної ІТ-компанії, яка підтримує діяльність провайдерів інфокомунікаційних послуг, надаючи їм відповідні компоненти і технології, у тому числі на основі моделі E2E.

Проєктування архітектури сервісу є складною інтелектуальною задачею, що не може бути вирішенна за допомогою формального її опису. Для автоматизації вирішення даної задачі можна застосувати методи та технології штучного інтелекту (ШІ) [8]. На даний час методи, технології та алгоритми ШІ, а особливо методи машинного навчання та нейронні мережі, успішно вирішують задачі з комп’ютерного зору [9], обробки та генерації природної мови [10, 11], перетворення звуку в текст та навпаки [12], генерації зображень та відео [13], в ситемах керування та прийняття рішень [14, 15] і т.д. Завдяки основним властивостям нейронних мереж, а саме універсальним апроксимаційним властивостям та здатності до навчання, вони перетворюються у інструмент високої точності для вирішення інтелектуальних задач. Формалізувавши вхідні та вихідні дані задачі, можна навчити нейронну мережу відображати зв’язок між цими даними за стратегією навчання з вчителем. Таким чином, можна вирішити задачу проєктування архітектури сервісів, які надаються провайдерам інфокомунікаційних послуг в інформаційних системах ІТ-компаній-розробників за моделлю E2E, за вимогами до даного сервісу.

Постановка проблеми

При обслуговуванні клієнтів провайдер інфокомунікацій надає їм визначені сервіси, намагаючись мінімізувати витрати, щоб можна було запропонувати найвигідніші умови на ринку [4]. Надання цих сервісів має бути інтегрованим в існуючу інфраструктуру провайдера. Оскільки загалом надається комплекс сервісів, то загальне рішення зі структури інтегрованого сервісу не видається простим. До того ж існує багато чинників, які впливають на це рішення. Тому побудова структури інтегрованого сервісу, яку ми називаємо архітектурою сервісу провайдера, перетворюється у складну проблему. Ми розглядаємо проблему побудови архітектури сервісу в умовах створення інформаційної системи провайдера на основі моделі E2E.

Існує загальна схема архітектури сервісу, яка визначає множину усіх можливих архітектур сервісів, які можна використовувати як проекти інформаційних систем провайдерів, які проектуються, реалізуються, впроваджуються і підтримуються на основі моделі E2E. Елементи, з яких можна будувати архітектуру для конкретного провайдера на основі цієї схеми, поділяються на такі класи:

1) Доменні високорівневі схеми;
2) Доменні схеми компонентів;
3) Компоненти доменні такі як успадковані компоненти провайдера, компоненти ІТ-компанії, компоненти третіх сторін;

4) Компоненти аналітики, ШІ і підтримки прийняття рішень;
5) Засоби взаємодії і комунікації;
6) Технічні засоби;
7) Засоби роботи з даними.

Особливістю компонентів ІТ-компанії та компонентів третіх сторін є те, що вони відповідають стандартам TMForum. Успадковані компоненти провайдера можуть не відповідати стандартам TMForum.

Проблема полягає у тому, щоб на основі інформації про провайдера згідно схеми розробити архітектуру інформаційної системи провайдера (архітектуру сервіса), яка оптимальним чином відповідає вимогам і очікуванням провайдера. Можна говорити про побудову персоналізованої під провайдера архітектури сервісу.

При цьому проблема вирішується у два етапи:

I. Побудова попередньої архітектури сервіса, яка передається провайдеру для аналізу, можливих змін і узгодження;

II. Побудова остаточної архітектури сервіса, яка буде затверджена і покладена в основу реалізації.

Ці етапи реалізуються різними нейромережами з глибоким навчанням, які викликає через REST API структурована LLM, яка знає особливості ув’язки і підтримки процесів життєвого циклу сервісів в інформаційних системах провайдерів, створених за моделлю E2E.

Загальний опис пропонованого рішення

Пропоноване рішення полягає у створенні інтелектуальної технології проектування архітектур сервісу на основі нейромереж з глибоким навчанням. Для створенні інтелектуальної технології проектування архітектур сервісів на основі нейромереж із глибоким навчанням, необхідно виконати такі кроки:

1) Виконати аналіз вхідних та вихідних даних, зібрати набір даних для навчання нейронної мережі;
2) Розробити методи формалізації вхідних та вихідних даних для нейронної мережі;
3) Розробити модель нейромережі для генерації архітектур сервісів;
4) Розробити метод навчання моделі нейромережі для генерації архітектур сервісів;
5) Розробити технологію автоматизованого проектування архітектури сервісів на базі запропонованої нейромережевої моделі.

Для побудови архітектури сервісів будемо використати підхід, побудований на реалізації і навчанні нейромережі з глибоким навчанням, яка, отримуючи на вході

характеристики провайдера і його вимоги до інформаційної системи – генерує архітектуру сервіса. Цей підхід пов’язаний з коцепцією життєвого циклу сервісів і передбачає тісне пов’язання і взаємне використання результатів етапів бізнес-аналізу, інженерії вимог і проектування (власне побудова архітектури сервісу).

Загалом дані етапів бізнес-аналізу, інженерії вимог потрібно підготувати, закодувати і у визначеному вигляді подати разом з закодованим представленням відповідних архітектур сервісів при навчанні нейромережі, а на вхід навченої нейромережі потрібно подавати тільки підготовлені дані етапів бізнес-аналізу, інженерії вимог для їх класифікації, тобто побудови відповідної архітектури сервісу для конкретного провайдера.

Аналіз вхідних, вихідних даних та їх формалізація

У даному розділі визначаються види і структура даних, що подаються на вхід нейронної мережі і видаються нейронною мережею на виході. Для цього спочатку розглядається питання формування набору даних. А потім для відібраних і структурованих розробляється система їх формального представлення та кодування для ефективного сприйняття нейронною мережею.

З огляду на сутність описаних проблеми і підходу до її вирішення мова йде про вибір і навчання нейронної мережі, яка, отримавши на вході характеристики провайдера і його вимоги до інформаційної системи, видає на виході архітектуру сервісу. При цьому потрібно проаналізувати приклади вимог провайдера до інформаційної системи – від самих мінімальних до максимальних – і відібрати ті з них, які мають реальний вплив на архітектуру сервісів. І потім треба навчити нейронну мережу підбирати і пов’язувати відповідним чином компоненти системи, будуючи правильну архітектуру.

Розпочнемо з вхідних даних. Природно, що на вхід нейронної мережі доцільно подавати характеристики провайдерів, їх діяльності та клієнтів. характеристики провайдерів, їх діяльності та клієнтів діляться на 6 груп характеристик:

1. Загальні характеристики бізнес-діяльності провайдерів.
2. Кількісні характеристики сервісів, що надаються провайдером за підтримки ІТ-компанії, та клієнтів).
3. Функціональні вимоги до компонентів системи.
4. Нефункціональні вимоги до сервісів (параметри SLA та ін.).
5. Специфічні вимоги до сервісів, які надаються провайдером.
6. Додаткові вимоги провайдера до сервісів.

Загальні характеристики бізнес-діяльності провайдерів виявляються на етапі бізнес-аналізу і пов'язані з визначенням цілей, організації, послуг, методів праці

і заінтересованих сторін провайдера. Можна передбачити вплив цих характеристик на вибір елементів архітектури, які забезпечують підрозділи і окремих виконавців організації інструментами для аналізу, прогнозування, та їх зв’язків з функціональними елементами архітектури.

Кількісні характеристики сервісів, що надаються провайдером за підтримки ІТ-компанії, та клієнтів загалом дозволяють оцінити навантаження на ресурси провайдера, пов'язане з наданням сервісу. Ці характеристики разом з характеристиками групи нефункціональних вимог до сервісів будуть впливатимуть на вибір тих елементів архітектури, які забезпечують умови, за яких функціональні елементи архітектури через їх взаємодію, управління доступом до ресурсів забезпечать інформаційній системі провайдера бажану поведінку.

Функціональні вимоги до компонентів системи традиційно поділятимемо на головні функції і функції в рамках головних. Їх визначення відбувається із застосуванням технік інженерії вимог на відповідному етапі створення інформаційної системи провайдера. Оскільки використовується мікросервісна архітектура, причому мікросервіси виділяються на основі доменного підходу, то в таблиці головні функції подаються через відповідні компоненти інформаційної системи. А потім визначаються функції компонентів, які використовує провайдер. Зрозуміло, що характеристики цієї групи визначально впливатимуть на вибір функціональних елементів архітектури, їх взаємозв’язків, будуть обумовлювати необхідність розширення функціональних можливостей мікросервісів.

Нефункціональні вимоги до сервісів визначаються у традиційний спосіб. Спочатку із широкого переліку нефункціональних вимог вибираються ті, які важливі для провайдера, а потім до них додаються кількісні показники, важливі для оцінювання рівня цієї вимоги до сервісу. Характеристики цієї групи традиційно впливатимуть на вибір елементів архітектури, які забезпечують умови, за яких функціональні елементи архітектури через їх взаємодію, управління доступом до ресурсів забезпечать інформаційній системі провайдера бажану поведінку.

Специфічні вимоги до сервісів, які надаються провайдером, стосуються застосовуваних для надання сервісу технологій, режимів, альтернативних варіантів. Вони впливають, з одного боку як і попередні дві групи на умови, за яких функціональні елементи архітектури через їх взаємодію, управління доступом до ресурсів забезпечать інформаційній системі провайдера бажану поведінку, з іншого боку, вони можуть додати елементи архітектури третіх сторін.

Додаткові вимоги стосуються компонентів третіх сторін, майстрів даних провайдера, окремих інструментів для виконавців, засобів інформування. Додаткові вимоги можуть додати до архітектури додаткові компоненти ІТ-компанії та компоненти третіх сторін.

Перейдемо до формального представлення та кодування вхідних даних для їх ефективного сприйняття нейронною мережею. Для подальшого використання і зручності формального опрацювання вимог до проектування сервісів визначимо їх як такий кортеж:

$$SARP = \left(N, G, Q, F, nF, S, A, M\right). \tag{1}$$

У формулі (1) прийняті такі позначення: *SARP* – ідентифікатор проєкту; *N* – ідентифікатор вимог (проєкту); *G* – загальні вимоги; *Q* – кількісні характеристики сервісів; *F* – функціональні вимоги; *nF* – не функціональні вимоги; *S* – специфічні вимоги до сервісів; *A* – додаткові вимоги; *M* – матриця подання архітектури сервісу.

Різноманіття вхідних даних супроводжується наявністю значної кількості форматів їх подання. Для функціональних вимог важливо вказати необхідність їх реалізації для провайдера чи відсутність такої потреби. Для інших груп важливими є різноманітні кількісні характеристики. Отже, змінні можуть приймати як бінарні значення так і кількісні в залежності від типу параметра.

Якщо вимога до архітектури сервісу по будь-якій характеристиці має бінарний характер то змінна приймає два значення, а саме 0 – вимога відсутня та 1 – вимога присутня. Якщо вимога має кількісних характер, вона приймає значення від 0 до 1, 0 – вимога відсутня та діапазон від 1 до 100%, що описує кількісні характеристики вимоги в відсотковому відношенні від мінімальних вимог до максимальних. В формальному представленні вимоги до архітектури сервісу перетворюються на одномірний масив. Кожний елемент масиву відображає блок та змінну даного блоку. Схематично подання вимог до архітектури сервісу представлено формулою (2):

$$N1 = \left(\underbrace{\underbrace{0}_{G_{1,1}} \; \underbrace{1}_{G_{1,j}} \; \underbrace{0}_{G_{2,1}} \; \underbrace{0.51}_{G_{3,j}} \; \underbrace{0.72 \; 0.42 \; 0.55 \; 0.23}_{G_{ij}} \; \ldots, \ldots, \ldots, \ldots, \ldots, \ldots}_{Q_{ij} \;\; F_{ij} \;\; nF_{ij} \;\; S_{ij}} \; \underbrace{0 \; 1 \; 0}_{A_{11,1}} \underbrace{}_{A_{ij}} \right) \tag{2}$$

У формулі (2) прийняті такі позначення:

*N*1 – ідентифікатор вимог(проєкту); G_{ij} – вимоги до загальних характеристик, що складаються з 12 блоків змінних G_i по декілька змінних в кожному блоці G_{ij}. Відповідно $G_{1,1} = 0$ – перша змінна з бінарним значенням, першого блоку загальних характеристик – клас провайдера за системою класифікації, $G_{2,1} = 0.51$ – перша змінна з кількісним значенням, другого блоку загальних характеристик – кількість користувальницьких сервісів провайдера і т.д.;

 ISSN 1560-8956

Q_{ij} – вимоги до кількісних характеристик сервісів архітектури, що проектується. Складається з 3 блоків змінних Q_i, кожен блок має від однієї до 15 змінних Q_{ij};

F_{ij} – вимоги до функціоналу архітектури, що проектується. Складаються з 19 блоків змінних F_i, що відповідає вимог до наявності функціонального блоку в архітектурі сервісу, по декілька змінних в кожному блоці F_{ij}, кожна змінна відповідає функціям даного блоку;

nF_{ij} – не функціональні вимоги до архітектури, що проектується. Складається з 10 блоків змінних nF_i, кожен блок має від однієї до декількох змінних nF_{ij};

S_{ij} – специфічні вимоги до сервісів. Складається з 5 блоків змінних S_i, кожен блок має по декілька змінних S_{ij};

A_{ij} – додаткові вимоги. Складається з 12 блоків змінних A_i, кожен блок має від однієї до декількох змінних A_{ij}.

Для визначення усіх можливих архітектур сервісу необхідно описати відповідний клас схем, який назвемо з**агальною схемою архітектури сервісу** – M**.** Структурування в з**агальній схемі архітектури сервісу** базується на традиційних для провайдерів телекомунікацій елементах архітектури, а саме компоненти та зв’язки між компонентами.

Загальна схема архітектури сервісу представлена квадратною матрицею $M_{n\times n}$, рядки і стовпці якої відповідають компонентам K_i, i=1,…,n, а елемент m_{ij} на перетині рядка i та стовпчика j визначає, чи використовує компонент K_i результати роботи компонента K_j і, якщо це так, то описує які параметри запиту компонент K_i передає компоненту K_j які дані компонент K_i отримує від компонента K_j. Загальний вигляд матриці наведений в таблиці 1 (тут елемент m_{ij}, $i=1,\ldots,n$, $j=1,\ldots,n$, якщо він не набуває значення *Null*, містить параметри запиту компонента K_i до компоненту K_j і дані, які компонент K_i надсилає компоненту K_j у відповідь).

Для того, щоб навчити нейронну мережу будувати архітектуру сервісу, яка в найбільш ефективний спосіб відповідає вимогам провайдера, важливо класифікувати компоненти. Це дозволить, з одного боку, пов’язати з кожним класом певний набір компонентів та їх зв’язків, з другого боку, уточнити зв’язки між компонентами різних класів у межах архітектури сервісу провайдера.

Надалі компоненти, які можна використовувати для побудови архітектури сервісу провайдера, поділимо на такі класи:

- успадковані компоненти провайдера;
- компоненти IT-компанії;

– компоненти третіх сторін;
– компоненти аналітики;
– ШІ і підтримки прийняття рішень;
– засоби взаємодії і комунікації;
– технічні засоби;
– засоби роботи з даними.

Таблиця 1.

Матриця подання архітектури сервісів

	K_1	…	K_j	…	K_n
K_1	m_{11}	…	m_{1j}	…	m_{1n}
…	…	…	…	…	…
K_i	m_{i1}	…	m_{ij}	…	m_{in}
…	…	…	…	…	…
K_n	m_{n1}	…	m_{nj}	…	m_{nn}

З огляду на постійні зміни у діяльності провайдерів цей перелік класів буде удосконалюватися у подальших дослідженнях.

Всі компоненти мають свій порядковий номер в множині компонентів *K*, від 1-го до 21-компонента. Номери компонентів є номерами рядків та стовпців квадратної матриці $M_{n\times n}$.

Щоб відобразити характер зв'язків між елементами архітектури сервісу будемо використовувати такі три стани:

– стан 1 - зв’язок між елементами відсутній, цей стан закодований числом 0;

– стан 2 – зв’язок між елементами присутній та обов’язковий, цей стан закодований числом 1;

– стан 3 – зв’язок між елементами присутній але він опційний, цей стан закодований числом 2.

Приклад закодованої архітектури сервісу в інформаційних системах провайдерів інфокомунікаційних послуг представлено у вигляді матриці у наведеній нижче формулі (3).

$$M_{nxn} = \begin{pmatrix} 0 & 1 & \dots & 2 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad (3)$$

 ISSN 1560-8956

Загалом у статті матриця $M_{n\times n}$ має 21 стовпець та 21 рядок, оскільки число можливих компонентів, які можна використовувати для побудови архітектури сервісу, складає 21. Зазначені 21 компонент використовуються у цій статті і при структуруванні нейронної мережі, і при її навчанні, тож і на виході нейронної мережі при її використанні отримуємо матрицю зазначеного розміру. Як було зазначено вище, з огляду на постійні зміни у діяльності провайдерів склад компонентів буде змінюватися, а отже у подальших дослідженнях перелік компонентів і розміри матриці будуть змінюватися.

Специфіка архітектури сервісу проявляється у тому, що описана формулою (3) матриця має діагональ, яка складається з нульових елементів. Іншими словами, у архітектурі сервісу не відображаються зв’язки компонентів самих з собою.

Для прикладу виберемо з’єднання компонента архітектури K_1 з компонентом K_2, між якими є обов’язковий зв’язок. Очевидно, цей зв’язок відображає елемент $m_{1,2} = 1$ матриці $M_{n\times n}$. Якщо якийсь елемент з множини можливих елементів K_n не має зв’язку з іншими елементами, то в архітектурі сервісу він не має бути представлений взагалі.

Розроблені в даному розділі метод формалізації опису вимог до архітектури сервісу (1) та метод формалізації самої архітектури сервісу (3) можуть бути використані як теоретична основа відповідно кодувальника вхідних даних для нейронної мережі і декодувальника вихідних даних нейронної мережі при автоматизованому проектуванні сервісів в інформаційних системах провайдерів інфокомунікаційних послуг. Також ці два методи використовуються у процесі навчання нейронної мережі генерувати архітектуру сервісу, виходячи з формалізованих вимог до неї.

Аналіз існуючих нейронних мереж для реалізації підходу до побудови архітектури сервісів

Для реалізації нейронної мережі з глибоким навчанням, призначеної для побудови архітектури сервісів, доцільно використати підхід генеративних нейромереж, як приклад для генерування зображень [12]. Входом нейронної мережі завжди є характеристика і вимоги провайдера, представлені у вигляді формули (2). Виходом нейронної мережі є матриця архітектури сервісу, представлена у вигляді формули (3). Найбільш перспективними моделями нейронної мережі як для генерації зображень, так і для генерації архітектури сервісу є дифузійні моделі [14] та трансформери з мультиувагою [15, 16].

Останнім часом згорткові нейронні мережі (CNN) стали центральним компонентом програм оброблення зображень у процесі розвитку нейронних мереж. Зорова система людини еволюціонувала до розрідженої та ефективної. Тому ми не

обробляємо все наше поле зору тими ж ресурсами, на відміну від звивин. Наші очі виконують стратегію точки фіксації за допомогою системи зорової уваги, яка відіграє важливу роль у людському пізнанні. Натхненні системою уваги в людському зорі, обчислювальні механізми уваги були розроблені та інтегровані в нейронні мережі. Однією з цілей їх впровадження є зменшення обчислювального навантаження, спричиненого операцією згортання. Варто також зазначити, що при цьому покращується продуктивність нейронної мережі.

Нейронні мережі на основі механізму уваги застосовувалися для створення різноманітних програм, наприклад, таких як розпізнавання зображень або відстеження об’єктів. Нова архітектура нейронної мережі кодера-декодера на основі уваги була представлена для нейронного машинного перекладу –Neural Machine Translation (NMT) у 2015 році. Ідея цього підходу проілюстрована на рис. 1, де показано, як працюють механізми уваги в нейронних мережах. У цьому прикладі кодер приймає вхідне речення англійською мовою, а декодер виводить його переклад голландською мовою. Кодер та декодер включають рекурентні нейронні мережі – Recurrent Neural Networks (RNN). Кодер виводить приховані стани, а декодер приймає всі приховані стани як вхідні дані. Перед їх обробленням декодер застосовує механізм уваги, який оцінює кожен прихований стан. Потім він множить кожен прихований стан на його оцінку, до якої застосовується функція активації softmax. Зважені оцінки підсумовуються, в результаті чого формується контекстний вектор для декодера. Шляхом отримання зважених прихованих станів, які найбільш пов’язані з певними словами, декодер фокусується на відповідних частинах вхідних даних під час декодування.

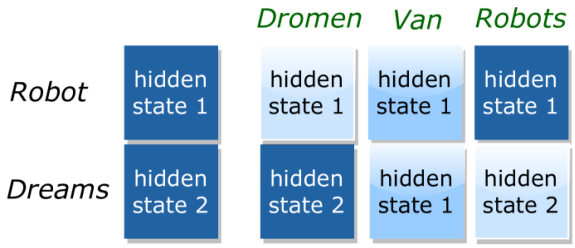

Рисунок 1. Приклад системи машинного перекладу
на основі нейронної мережі

Механізми уваги в нейронних мережах швидко прогресували, особливо для NMT. Одним з них є механізм самоуваги, який є основним будівельним блоком нейронних мереж трансформерів. Оригінальна нейронна мережа трансформер складається зі стеків кодерів-декодерів, які повністю базуються на механізмі самоуваги без будь-яких згорткових або рекурентних шарів. У кожному зі стеків

 ISSN 1560-8956

кодера-декодера, які утворюють трансформер, є шість ідентичних шарів. Щоб проілюструвати модель, на рис. 2 показаний лише один стек кодера-декодера.

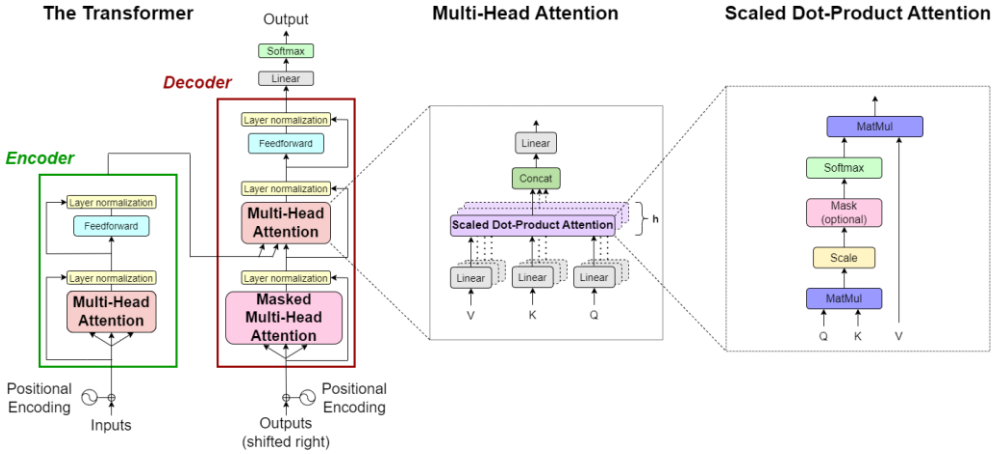

Рисунок 2. Деталі архітектури нейронної мережі трансформер
(Зліва) Трансформер з одним стеком кодера-декодера.
(У центрі) Механізм багатоголової уваги.
(Праворуч) Механізм уваги скалярного добутку.

Стеки кодера-декодера в трансформері складаються з повністю пов’язаних шарів і механізму багатоголової уваги, яка є свого роду механізмом самоконтролю. Механізм самоконтролю застосовує масштабований скалярний добуток уваги всередині себе. Як видно на рис. 2, ці механізми уваги використовують три вектори для кожного слова, а саме запит (Q), ключ (K) і значення (V). Ці вектори обчислюються шляхом множення вхідних даних на вагові матриці W_q, W_k і W_v, які налаштовуються під час навчання. Загалом, кожне значення зважується функцією запиту з відповідним ключем. Вихідні дані обчислюються як зважена сума значень. У масштабованому скалярному добутку обчислюється скалярний добуток запиту з усіма ключами. Як зазначено в (4), кожен результат ділиться на квадратний корінь із розмірності ключів (щоб мати більш стабільні градієнти) і передається у функцію *softmax*. Нарешті, кожна оцінка softmax множиться на значення:

$$Attention\left(Q,K,V\right)=soft\max\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

Багатоголова увага розширює цю ідею, застосовуючи лінійні активації до вхідних даних (ключів, значень і запитів) h разів на основі різних вивчених лінійних представлень (див. рис. 2). Кожна з спроектованих версій запитів, ключів і значень називається заголовками, для яких паралельно виконується масштабований скалярний добуток. Таким чином, самоувага обчислюється кілька разів з використанням різних наборів векторів запиту, ключів і значень. Це дозволяє моделі спільно обробляти

інформацію на різних позиціях. На останньому кроці проекції з’єднуються. Крім того, декодер застосовує замасковану увагу кількох голів, щоб переконатися, що під час прогнозування наступного слова в реченні використовуються лише попередні вбудовані слова (токени).

Існують різні архітектури нейронних мереж трансформерів для різних завдань. Після того як трансформери досягли значного прогресу в обробленні природної мови, вони були адаптовані для виконання завдань оброблення зображень. На рисунку 3 показана одна з таких адаптацій, у якій трансформер використовується для обробки зображень. Він застосовує самоконтроль у локальних околицях для кожного пікселя запиту та добре працює при створенні зображень і суперроздільності зображень. Поточною сучасною моделлю є Vision Transformer (ViT) [17-19], який розбиває вхідне зображення на фрагменти перед тим, як Transformer візьме лінійне вбудовування цих фрагментів у послідовність як вхідні дані (рис. 3).

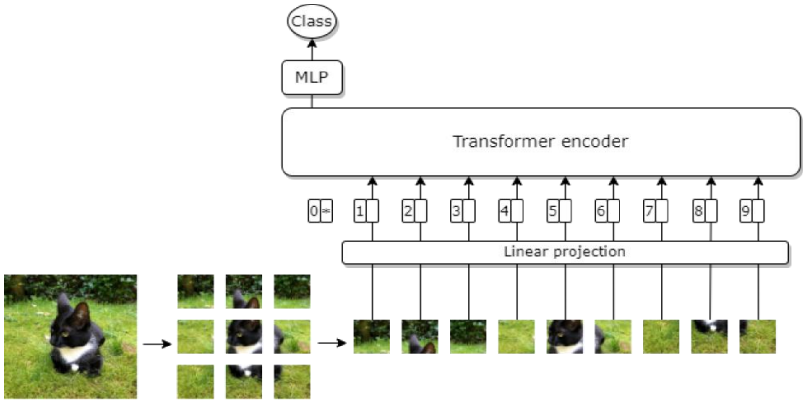

Рисунок 3. Трансформер для обробки зображень

ViT добре справляється із завданнями класифікації зображень, використовуючи при цьому менше обчислювальних ресурсів. Тому сьогодні моделі ViT широко застосовуються для вирішення завдань оброблення зображень у різних галузях діяльності людини.

Також сьогодні завдяки моделям перетворення тексту в зображення, таким як DALL-E 3 або Midjourney, ШІ перетворився на популярний інструмент для створення захоплюючих зображень. Дійсно, дифузійні моделі виявилися дуже ефективними у створенні високоякісних зображень та перевершили при цьому у синтезі зображень відому модель GAN [14].

На відміну від GAN, навчання дифузійних моделей не потребує змагальності. Оригінальний метод шумозаглушення був натхненний нерівноважною термодинамікою, яка систематично руйнує структуру в розподілі даних, а потім відновлює дані. На основі цього методу ймовірнісні моделі дифузії з усуненням шуму були застосовані для створення зображень.

Дифузійні моделі потребують двох основних етапів у фазі навчання, зображених на рис. 4. На першому з них, під час прямого (дифузійного) процесу, до вхідного зображення поступово додається випадковий шум, доки оригінальний вхід не стане повністю шумом. Така трансформація подається за допомогою фіксованого ланцюга Маркова, який додає гаусівський шум для T послідовних кроків. На другому етапі, під час процесу реконструкції або зворотного процесу модель реконструює вихідні дані з шуму, отриманого в прямому процесі. Зворотний процес подається як ланцюг Маркова з навченими переходами Гауса. Відповідно, передбачення щільності ймовірності в момент часу t залежить тільки від стану, досягнутого в момент часу $(t-1)$. Тут $x_1,\ldots,x_T$ є латентами тієї ж розмірності, що й дані, які роблять моделі дифузії моделями латентної змінної.

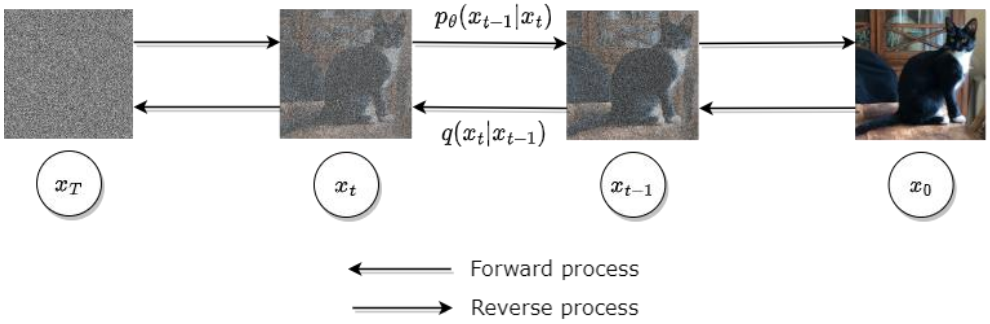

Рисунок 4. Принцип роботи моделі дифузії в цілому

Загальна структура дифузійної моделі наведена на рис. 4. Оцінювання щільності ймовірності на більш ранньому відрізку часу, враховуючи поточний стан системи, є нетривіальною задачею, тому зворотний процес вимагає навчання нейронної мережі. З цією метою всі попередні градієнти необхідні для отримання необхідної оцінки. Для кожного кроку в ланцюзі Маркова нейронна мережа вчиться знімати шуми в зображенні. За бажанням, процес усунення шумів можна керувати текстом. У цьому випадку кодер Transformer відображає текстову підказку на токени, які потім передаються в нейронну мережу (див. рис. 5). Після навчання дифузійну модель можна використовувати для генерування даних шляхом простого проходження випадкового шуму (і додатково текстової підказки) через навчений процес усунення шумів. Слід зазначити, що цей процес потребує кількох кроків усунення шумів при застосуванні моделі. Відповідно, дифузійні моделі повільно генерують зображення, коли з’являється запит. У результаті було запропоновано кілька методів для прискорення отримання результату від багатоетапного до кількох- і навіть одно-крокового. Наприклад, поточний метод SOTA для одноетапного та малоетапного висновку – Adversarial Diffusion Distillation використовує тренування змагальності,

щоб підштовхнути генерацію зображень до розумних зображень із меншою кількістю кроків усунення шумів.

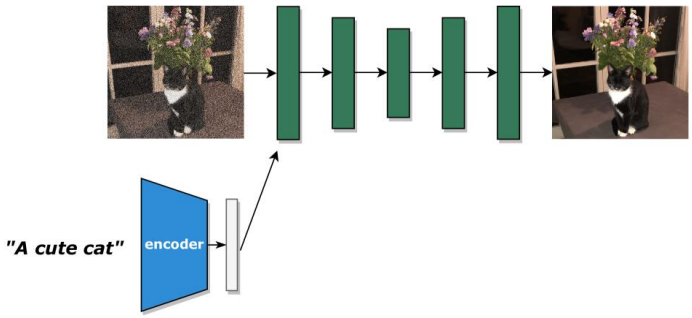

Рисунок 5. Ілюстрація одного часового кроку
вивченого ланцюга Маркова у зворотному процесі

Тут глибока нейронна мережа вчиться перетворювати зашумлений вхідний сигнал у менш зашумлене зображення за допомогою текстової підказки, яка описує вміст зображення.

Дифузійна модель може здатися подібною до VAE, які кодують вхідні дані як розподіл імовірностей, а потім вибірку з отриманого прихованого простору. Однак прямий процес робить дифузійну модель відмінною від VAE, оскільки навчання в цьому фіксованому ланцюгу Маркова не потрібне.

Навчання нейромережі

Навчання нейромережевої моделі відбувається за стратегією навчання з вчителем. При навчанні з учителем нейронна мережа навчається на розміченому наборі даних і прогнозує відповіді, які використовуються для оцінки точності алгоритму на навчальних даних. Навчання з учителем передбачає наявність повного набору розмічених даних для навчання моделі. Наявність повністю розміченого набору даних означає, що кожному входу в навчальному наборі відповідає результат, який алгоритм і повинен отримати.

Отже, завдання алгоритму навчання полягає у підборі таких параметрів моделі, при яких буде отримано мінімальну сумарну середньоквадратичну різницю між поточними виходами моделі та заданими виходами – вчителем, при заданих вхідних значеннях.

Для навчання використовуються приклади проектів SARP в яких є відповідні пари. Парами вхід і цільовий вихід для нейронної мережі є вимоги до проекту N, а цільовим виходом є архітектура, задана матрицею M. Таким чином, набір даних, що складається з прикладів *SARP*, становить собою повністю розмічений набір даних, який застосовується в процесі реалізації стратегії навчання з вчителем.

На вхід мережі подаюся вимоги до архітектури сервісу які кодуються в одномірну матрицю за формулою (2). Функція, яка реалізує кодування вимог до

архітектури сервісу, називається токенізатором вимог до сервісу. Результатом роботи функції токенізації вимог до архітектури сервісу є одновимірний масив, який складається з 441 елемента, кожен з яких вказує на відповідну вимогу до архітектури.

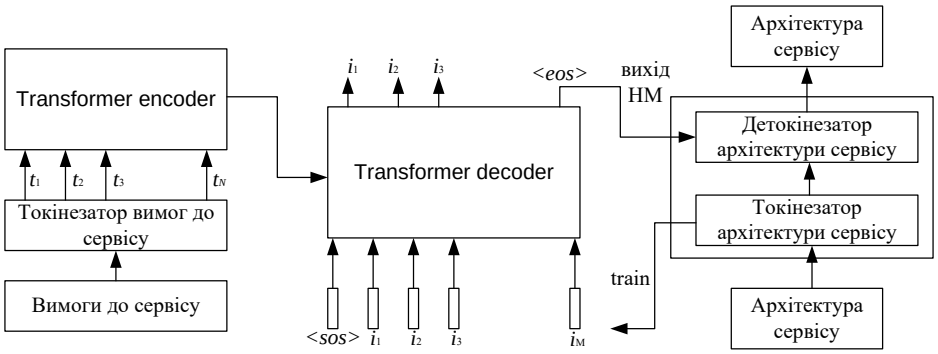

Рисунок 6. Ілюстрація процесу навчання нейронної мережі генерації архітектури сервісу

Таким чином, функція токенізації архітектури сервісу кодує архітектуру сервісу з навчального набору даних в матрицю, для навчання нейронної мережі за стратегією навчання з вчителем. Метод кодування архітектури сервісу в матрицю *M* описано таблицею 1 та формулою (3), в результаті отримується описана вище квадратна матриця розміром 21 на 21 елемент, яка описує компоненти сервісу та їх зв’язки.

```
Значення за умовою (Y_data):
tensor([[0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 1, 2, 1, 2, 2, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [2, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Рисунок 7. Задана архітектура сервісу для навчання нейронної мережі

```
Передбачено (Y_data):
tensor([[0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 1, 2, 1, 2, 2, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [2, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Рисунок 8. Результат синтезу архітектури
сервісу навченою нейронною мережею

На виході з нейронної мережі маємо матрицю, що описує архітектуру сервісу яка за допомогою окремої функції детокінезації архітектури сервісу декодується в архітектуру сервісу, процес обернений до методу.

Проведено навчання нейронної мережі набором даних що складається з проектів *SARP,* втрати нейромережі становили 0.000014. Приклад заданої архітектури сервісу для навчання нейронної мережі, в якості вчителя, показано на рис. 7.

Приклад синтезу архітектури сервісу в вигляді матриці подання архітектури сервісу навченою нейронною мережею показано на рис. 8. Як видно з рис. 7 та рис. 8, нейронна мережа генерує матрицю подання архітектури сервісу *M* зі 100% точністю.

Навчання нейромережі прикладами проектів *SARP* дозволяє виконати пошук спільних особливостей діяльності провайдерів, для яких побудовані архітектури зі спільними високорівневими схемами або доменними схемами компонентів або однаковими компонентами і зв’язками.

Після процедури навчання нейронної мережі, навчена модель інтегрується в інформаційну систему автоматизованого проектування архітектури сервісу по заданим вимогам. Розробник архітектури збирає вимоги та завдає їх в системі для проектування. Система кодує вимоги токенізатором вимог до сервісу перетворюючи дані в тензори, що надходять на навчену нейронну мережу. Нейронна мережа генерує на виході тензор, який потім окремою функцією декодується в архітектуру сервісу. Процес роботи системи проектування показана рис. 7. Файл зі згенерованою архітектурою сервісу далі відкривається як проект в спеціалізованій програмі і доступний для подальшого коригування та доопрацювання інженерами.

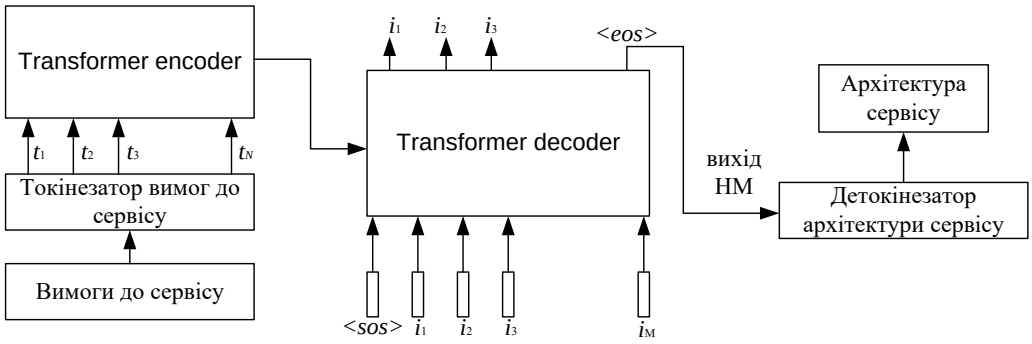

Рисунок 7. Ілюстрація процесу навчання
нейронної мережі генерації архітектури сервісу

В процесі подальшої роботи, нові і скориговані інженерами архітектури сервісу додаються до навчального набору даних. При додаванні певної кількості нових прикладів, доопрацьованих інженерами, нейронна мережа перенавчається для корекції її більш правильної роботи.

Висновки

У даній статті розглянута проблема побудови архітектури сервісів в інформаційних системах провайдерів інфокомунікаційних послуг. Запропоновано підхід до побудова архітектури сервісів на основі нейронної мережі з глибоким навчанням у інформаційних системах провайдерів інфокомунікаційних послуг.

Це дозволяє будувати архітектури сервісу з урахуванням найбільш характерних параметрів діяльності провайдерів і особливостей їх клієнтів, функціональних, часових і фінансових вимог та інших важливих чинників впливу. Навчена у такий спосіб нейронна мережа буде здатна використовувати накопичений досвід побудови ефективної архітектури сервісів.

Виконане експериментальне дослідження показало придатність побудованих архітектур, побудованих за допомогою навченої нейромережі. Експериментальні результати свідчать, що навчена нейромережа з глибоким навчанням будує архітектури, які у незначній пропорції можуть поліпшити досвідчені архітектори сервісів.

На основі запропонованого підходу можна будувати архітектуру з урахуванням накопиченого досвіду, що робить доцільним його використання у платформі підтримки життєвого циклу сервісів, орієнтованій на використання у інформаційних системах провайдерів інфокомунікаційних послуг.

Подальші дослідження пов’язані з розвитком і доповненням набору даних для навчання нейронної мережі, що дуже важливо для розвитку технології побудови архітектури сервісів.

Список використаних джерел

1. Saraiva de Sousa, N. F., Lachos Perez, D. A., Rosa, R. V., Santos, M. A. S., & Esteve Rothenberg, C. Network Service Orchestration: A survey. Computer Communications. – 2019 – Vols. 142–143. – pp. 69-94. https://doi.org/10.1016/j.comcom.2019.04.008

2. В. Чимшир, С. Теленик, О. Ролік, Е. Жаріков. Платформа підтримки життєвого циклу сервісів в інформаційних системах провайдерів інформаційно-комунікаційних послуг. Адаптивні системи автоматичного управління: міжвідомчий науково-технічний збірник. – 2023. – № 1 (42). – С. 205-226. https://doi.org/10.20535/1560-8956.42.2023.279172

3. Manvi, S. S., & Krishna Shyam, G.. Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. Journal of Network and Computer Applications. – 2014. – Vol. 41. – pp. 424-440. https://doi.org/10.1016/j.jnca.2013.10.004

4. Widianto, A., & Subriadi, A. P. IT service management evaluation method based on content, context, and process approach: A literature review. Procedia Computer Science. – 2022. – Vol. 197. – pp. 410-419. https://doi.org/10.1016/j.procs.2021.12.157

5. Yu, Q., Zhao, N., Li, M., Li, Z., Wang, H., Zhang, W., Sui, K., & Pei, D. A survey on intelligent management of alerts and incidents in IT services. Journal of Network and Computer Applications. – 2024. – Vol. 224. – №103842. https://doi.org/10.1016/j.jnca.2024.103842

6. Sun, R., & Gregor, S. Reconceptualizing platforms in information systems research through the lens of service-dominant logic. In The Journal of Strategic Information Systems. – 2023. – Vol. 32(3). – №101791. https://doi.org/10.1016/j.jsis.2023.101791

7. Zakaria, A. F., Lim, S. C. J., & Aamir, M. A pricing optimization modelling for assisted decision making in telecommunication product-service bundling. In International Journal of Information Management Data Insights. – 2024. – Vol. 4(1) . – №100212. https://doi.org/10.1016/j.jjimei.2024.100212

8. Collins, C., Dennehy, D., Conboy, K., & Mikalef, P. Artificial intelligence in information systems research: A systematic literature review and research agenda. International Journal of Information Management. – 2021. – Vol. 60. – №102383. https://doi.org/10.1016/j.ijinfomgt.2021.102383

9. K. Khotin, V. Shymkovych, P. Kravets, A. Novatsky, L. Shymkovych. Convolutional neural network for dog breed recognition system. Адаптивні системи автоматичного управління: міжвідомчий науково-технічний збірник. – 2024. – № 2 (45). – С. 3-14. https://doi.org/10.20535/1560-8956.45.2024.313022

10. Steblianko, O., Shymkovych, V., Kravets, P., Novatskyi, A., & Shymkovych, L. Scientific article summarization model with unbounded input length. Information.

Computing and Intelligent systems. – 2024. – Vol. 5. – pp. 150-158. https://doi.org/10.20535/2786-8729.5.2024.314724

11. Kobchenko, V.R., Shymkovysh, V.M., Kravets, P.I., Novatskyi, A.O., Shymkovysh, L.L., & Doroshenko, A.Y. An intelligent chatbot for evaluating the emotional colouring of a message and responding accordingly. PROBLEMS IN PROGRAMMING. – 2024. – Vol. 1. – pp. 23-29. http://doi.org/10.15407/pp2024.01.23

12. Ahlawat, H., Aggarwal, N., & Gupta, D. Automatic Speech Recognition: A survey of deep learning techniques and approaches. International Journal of Cognitive Computing in Engineering. – 2025. – Vol. 6. – pp. 201-237. https://doi.org/10.1016/j.ijcce.2024.12.007

13. Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., Hutchinson, B., Han, W., Parekh, Z., Li, X., Zhang, H., Baldridge, J., & Wu, Y. (2022). Scaling Autoregressive Models for Content-Rich Text-to-Image Generation (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2206.10789

14. M. Osypenko, V. Shymkovych, P. Kravets, A. Novatsky, L. Shymkovych. Intelligent control system with reinforcement learning for solving video game tasks. Адаптивні системи автоматичного управління: міжвідомчий науково-технічний збірник. – 2024. – № 2 (45). – С. 34-46. https://doi.org/10.20535/1560-8956.45.2024.313065

15. Kravets, P., Novatskyi, A., Shymkovych, V., Rudakova, A., Lebedenko, Y., Rudakova, H. Neural Network Model for Laboratory Stand Control System Controller with Parallel Mechanisms. Lecture Notes on Data Engineering and Communications Technologies. – 2023. – Vol. 181. – pp. 47-58. https://doi.org/10.1007/978-3-031-36118-0_5

16. Wang, X., He, Z., & Peng, X. Artificial-Intelligence-Generated Content with Diffusion Models: A Literature Review. Mathematics. – 2024. – Vol. 12(7). – №977. https://doi.org/10.3390/math12070977

17. Islam, S., Elmekki, H., Elsebai, A., Bentahar, J., Drawel, N., Rjoub, G., & Pedrycz, W. A comprehensive survey on applications of transformers for deep learning tasks. Expert Systems with Applications. – 2023. – Vol. 241. – №122666. https://doi.org/10.1016/j.eswa.2023.122666

18. Le, D. P. C., Wang, D., & Le, V.-T. A Comprehensive Survey of Recent Transformers in Image, Video and Diffusion Models. Computers, Materials & Continua. – 2024. – Vol. 80(1). – pp. 37-60. https://doi.org/10.32604/cmc.2024.050790

19. Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., Yang, Z., Zhang, Y., & Tao, D. A Survey on Vision Transformer. IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2023. – Vol. 45(1) . – pp. 87-110. https://doi.org/10.1109/tpami.2022.3152247