

АДАПТИВНІ ЗАСОБИ ЗАХИСТУ КОМП'ЮТЕРНИХ СИСТЕМ НА ОСНОВІ АПАРАТА НЕЙРОННИХ МЕРЕЖ

Коновал В.О.¹, Мухін В.Є.²

Національний технічний університет України «Київський політехнічний інститут
ім. Ігоря Сікорського», Київ, Україна

¹ konoval.volodymyr@lll.kpi.ua, ² v_mukhin@i.ua [0000-0002-1206-9131]

В останні роки проблема виявлення та реагування на зміни в даних в режимі реального часу стає все більш актуальною. Дана проблема полягає в тому, що з часом дані можуть змінити свій розподіл і ті класифікатори даних, які раніше показували високу точність роботи, можуть почати видавати неправильні оцінки цих даних або мітки класів. Дана проблема носить назву дрейф концепції. Методів виявлення дрейфу а також правильного реагування на нього за останні роки було розроблено немало, але всі ці методи мають ряд недоліків. Серед цих недоліків можна виділити необхідність доступу до справжніх міток класів, що в умовах реального світу є доволі витратною по ресурсам задачею, а також високий рівень помилкових спрацьовувань. В даній роботі буде розглянуто підхід до виявлення дрейфу концепції через використання невпевненості ансамблю глибоких нейронних мереж. Даний метод для виявлення дрейфу концепції не вимагає відомостей про значення істинних міток класів.

Ключові слова: дрейф концепції, штучні нейронні мережі, штучні імунні системи, адаптація, потоки даних, адаптація до дрейфу концепції, адаптація нейронних мереж

1. ВСТУП

Область класифікації даних є доволі актуальною та широкою галуззю застосування машинного навчання і особливо нейронних мереж. В даній класифікації на вхід ми отримуємо на вхід набір даних і натренована нейронна мережа повинна класифікувати ці дані і віднести їх до тієї чи іншої групи. В області безпеки дана галузь можна активно застосовувати для класифікації обличч людей на основі зйомок камер відеоспостереження або класифікації трафіку мережі і визначення, чи не було спроби проведення атак на мережі, такі як наприклад ХСС атаки чи атака сканування портів. Але з часом злочинці розроблюють нові методи атак, які нейронній мережі є невідомим. Зміна цих атак може в свою чергу впливати на розподіл даних, які нейронна мережа отримує для їх подальшої класифікації. Зміна розподілу даних, яка призводить до неправильної класифікації даних називається дрейф концепції. Для виявлення даного дрейфу було розроблено ряд алгоритмів виявлення та реагування на його різні типи але дані методи мають ряд недоліків, такі як необхідність доступу до дійсних значень класу даних, що в реальних умовах вимагає постійної присутності спеціалістів для розміток даних, а також, в деяких випадках, велика кількість помилкових спрацьовувань.

В даній роботі буде розглянутий новий методи виявлення дрейфу концепції, який використовує невизначеність ансамблю глибоких нейронних мереж. Даний метод розраховує невизначеність класифікаторів ансамблю відносно отриманих даних і використовує дану невизначеність для виявлення дрейфу концепції.

2. ДРЕЙФ КОНЦЕПЦІЇ

Поняття «дрейф концепції» використовується для опису будь-яких змін у ймовірнісному розподілі даних в потоці даних, тобто, джерело концепції даних змінюється після певних проміжків часу, що в свою чергу приводить до погіршення класифікації моделі. При виникненні дрейфу концепції, навчальна модель, яка натренована на попередніх даних, може стати неефективною для нових даних, які були взяті з іншого розподілу. Для уникнення впливу дрейфу концепції на точність передбачення, алгоритми роботи з потоками даних повинні адаптуватися до даної нестабільності в потоках даних. Існують два типи алгоритми роботи з потоками даних, які працюють з нестационарними середовищами: явні та неявні.

Явний алгоритм виявлення зміни концепції виявляють появу дрейфу концепції та адаптують модель до даного дрейфу в залежності від його типу. Неявний алгоритм, який також можуть називати пасивним, не виконують процес визначення появи дрейфу а доволі просто проводять процес оновлення моделі навчання з плином часу з певною постійною швидкістю. Очевидним недоліком неявного підходу є повільна адаптація до концепцій, що змінюються, без явних відомостей про фактичний дрейф концепції.

Сам же дрейф концепції можна поділити на реальний та віртуальний дрейф. Віртуальним дрейфом називають дрейф, коли зміни в апіорній ймовірності простору властивостей ($P(X)$) не впливає на межу прийняття рішень. На противагу віртуальному дрейфу концепції, реальний дрейф концепції вказує на зміни у ймовірності вихідних класів (y) для заданого простору входу (X) ($P(y|X)$), що в результаті призводить до зміни межі прийняття рішень.

Також реальний дрейф концепції може виникати і без зміни ймовірності простору властивостей $P(X)$.

Крім розподілу на віртуальний та реальний, дрейф концепції може відбуватися з різною частотою, яких згідно до роботи Gulcan та Can [1] можна розділити на чотири категорії: раптовий, поступовий, інкрементальний та періодичний.

- Раптовий дрейф вказує на раптову і неочікувану зміну в розподілі даних, тобто, в певний момент часу t розподіл даних раптово змінюється іншим розподілом, що призводить до гострого погіршення результатів класифікації.
- Поступовий дрейф концепції відбувається в процесі, коли кількість прикладів з нового розподілу поступово збільшується а кількість прикладів зі старого розподілу поступово зменшується, що в свою чергу призводить до поступового збільшення ймовірності вибірки нового розподілу разом зі зменшенням ймовірності вибірки попереднього розподілу.
- Інкрементальний дрейф концепції відбувається, коли в певній частині області властивостей відбуваються незначні зміни, що з плином часу призводить до поступової зміну розподілу на новий. Даний тип дрейфу відрізняється від поступового тим, що зміна розподілу відбувається за набагато більший проміжок часу, ніж при поступовому дрейфі і відповідно такий дрейф концепції важче відслідкувати в потоці даних.
- Періодичний або повторюваний дрейф концепції вказує на повторну появу деяких вже зустрінутих розподілів концепції після певного проміжку часу.

3. МЕТОДИ ВИЗНАЧЕННЯ ДРЕЙФУ КОНЦЕПЦІЇ

Дослідженню методів визначення дрейфу концепції в потоках даних присвячено велика кількість наукових робіт. Серед розроблених методів найбільш відомими та такими, що найчастіше використовуються в інших дослідницьких роботах, можна виділити методи розроблені в роботах [2–5], які будуть коротко розглянуті далі. Варто зауважити, що дані алгоритми визначають саме наявність дрейфу концепції в певному наборі даних, а визначення підтипу дрейфу та правильна адаптація до них буде розглянуто у наступному підрозділі.

Одним з основних методів визначення дрейфу концепції можна назвати метод, розроблений та продемонстрований в дослідженні [2]. В даній роботі автори продемонстрували розроблений ними метод детекту дрейфу (Drift Detection Method, DDM), який використовує рівень помилки навчальної системи вважаючи, що рівень помилки системи буде зменшуватись з кількістю проаналізованих прикладів при умові, що розподіл даних залишається стаціонарним. Якщо алгоритм помічає, що рівень помилки перевищує певне порогове значення, то він інформує що або відбувся дрейф концепції або він відбудеться у найближчому майбутньому.

В свою чергу в дослідженні [3] було представлено модифікацію методу DDM, яка носить назву EDDM (Early Drift Detection Method). Даний метод, як і метод DDM, для визначення дрейфу концепції використовує помилки точності класифікації, але EDDM замість врахування рівня помилок враховує відстань між двома помилками ведучи підрахунок стандартного відхилення та максимальної відстані і максимального стандартного відхилення. Даний метод краще себе демонструє в процесі визначення поступового дрейфу концепції при цьому також показуючи гарні результати при виникненні різкого дрейфу.

Автори дослідження [4] представили метод визначення дрейфу з використанням ковзаючого вікна під назвою Adaptive Windowing (ADWIN). Даний алгоритм використовує ковзаючі вікна, розмір яких визначається в реальному часі у відповідності до швидкості змін даних в самому вікні. Даний підхід зручний завдяки тому, що користувачеві не потрібно самому визначати параметри вікна або розмір кроку. Результати експериментів показали, що даний підхід показує кращі результати роботи ніж методи з вікнами фіксованого розміру. Тестування ефективності роботи класифікатора проводилося як на штучних так і реальних датасетах і показало доволі високі результати демонструючи низькі показники хибнопозитивних та хибнонегативних показників класифікації.

В дослідженні [5] було запропоновано метод визначення дрейфу концепції, що заснований на використанні статистичного тесту Колмогорова-Смірнова та ковзаючого вікна фіксованого розміру. Розроблений метод під назвою Kolmogorov-Smirnov Windowing, коротко KSWIN не вимагає представлення розподілу даних для своєї роботи, він може моніторити дані або розподіл продуктивності роботи для детекту дрейфу концепції. Результати експериментів на штучних датасетах демонструють кращі середні результати точності визначення дрейфу концепції у порівнянні з раніше розглянутими методами визначення дрейфу концепції: DDM, EDDM, ADWIN.

4. ВИКОРИСТАННЯ НЕВИЗНАЧЕНОСТІ МЕРЕЖІ ДЛЯ ВИЯВЛЕННЯ ДРЕЙФУ КОНЦЕПЦІЇ

Існує ряд підходів до визначення невпевненості моделі, одним з даних підходів є підхід відсіву Монте-Карло. Даний підхід полягає у застосуванні шару відсіву між внутрішніми шарами нейронної мережі. Даний шар відключає частину нейронів на одному внутрішньому шарі і в першу чергу застосовується для запобігання перенавчання моделі. При багаторазовій подачі одного прикладу у нейронну мережу з даними шарами відсіву ми отримаємо масив передбачень, які можуть між собою відрізнятися у том, оскільки шар відсіву щоразу буде відмикати різні нейрони шару. Варто зауважити, що різниця в результатах передбачень для прикладу можлива лише в тому випадку, якщо нейронна мережа невпевнена відносно цих даних, оскільки при класифікації відомих даних, розбіжність у передбаченнях буде вкрай малою або буде відсутня взагалі. Далі використовуючи даний масив даних ми можемо отримати усередненні передбачення моделі відносно певного набору даних а потім розрахувати ентропію Шенона для даного набору, що і буде являти собою невпевненість моделі відносно класифікованих даних.

Підхід, розроблений в даній роботі, замість розрахунку невизначеності з використанням відсіву Монте-Карло, використовує передбачення ансамблю глибоких класифікаторів. Причин для використання ансамблю класифікаторів декілька. Перша причина полягає в тому, що ансамбль класифікаторів показує вищу точність передбачень у порівнянні з одним класифікатором. По друге, працюючи з ансамблем у випадку перенавчання одного його елементів, його можна легко замінити новим класифікатором, який буде навчений на тих даних, на яких в свій час тренувалися початкові класифікатори з набору.

Сам процес класифікації потоку даних буде відбуватися наступним чином:

1. З потоку даних виділяється набір даних фіксованого розміру, в даному випадку розміром 1000, і надсилається ансамблю на класифікацію. Якщо кількість даних в потоці менша за фіксований розмір, то надсилається ті, що залишилися без очікування появи нових даних.

2. Кожен з класифікаторів ансамблю робить власні передбачення відносно отриманого набору даних, які збираються в масив.

3. Отримані дані використовуються для розрахунку передбачень ансамблю і розрахунку невизначеності ансамблю. Передбачення ансамблю розраховуються наступною формулою:

$$p_{ens}(x) = \frac{1}{N} \sum_{i=1}^N p_i(x),$$

де

x – набір вхідних даних;

$p_i(x)$ – передбачення окремо взятого класифікатора відносно набору даних x ;

N – кількість класифікаторів у ансамблі.

Невизначеність же ансамблю розраховується як ентропія Шенона і має наступний вигляд:

$$u_{ens}(x) = - \sum_{i=1}^N p_i(x) * \log_2(p_i(x)).$$

4. Отримана невизначеність подається на вхід алгоритму виявлення дрейфу концепцій ADWIN який розбиває отримані дані на частини і розраховує їх усередненні значення. І у випадку перевищення цих значень встановленого порогу, то алгоритм повідомляє про виникнення дрейфу

5. У випадку виникнення дрейфу концепції відбувається донавчання ансамблю на даних до його виникнення, перезапуск алгоритму ADWIN і встановлення індексу, на якому виник дрейф як такого, починаючи з якого будуть виділятися нові партії даних з потоку і перехід до пункту 1.

6. Якщо дрейф концепції не виник, перехід до пункту 1.

5. РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

В якості датасетів для досліджень було обрано ряд штучних датасетів, які були згенеровано спеціально з дрейфами концепцій, та ряд реальних датасетів. Дане рішення було прийняте у зв'язку з тим, що реальні датасети можуть не містити в собі дрейфи концепцій або бути недостатньо наочними. Порівняння буде проводитися між розробленим алгоритмом, тобто алгоритмом використання невизначеності ансамблю і методом ADWIN, методом ADWIN з використанням помилки ансамблю на даних а також методом KSWIN.

Розглянемо результати роботи алгоритму на одному з наборів даних.

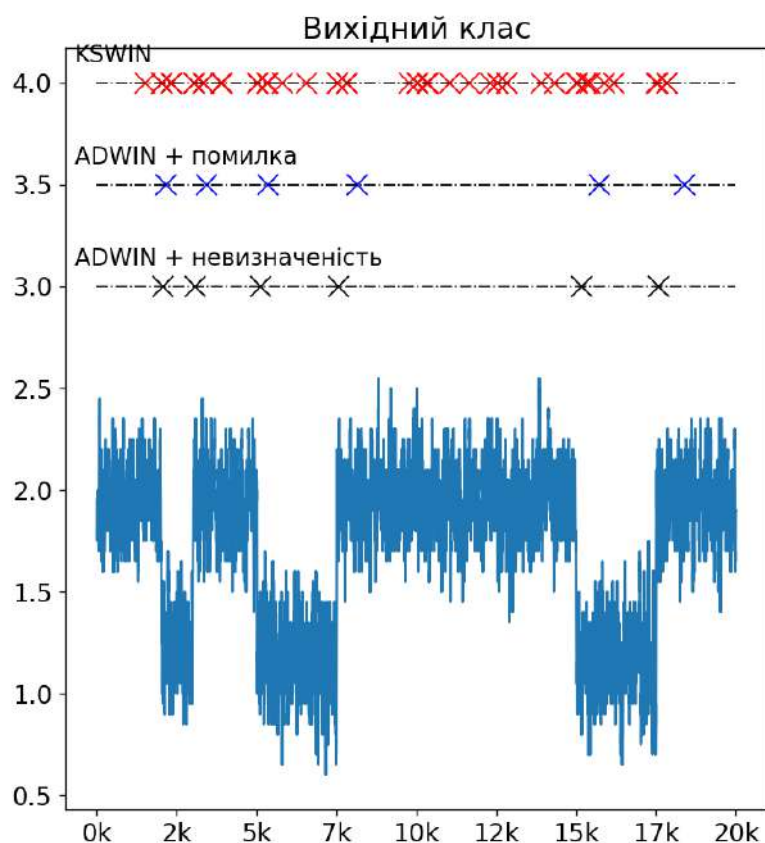


Рисунок 1. Приклад роботи алгоритмів

На даному рисунку зображено приклад роботи алгоритмів виявлення дрейфу концепцій. Чорні мітки відповідають тим місцям, де розробленим в даній роботі алгоритмом, червоним місця дрейфу, визначені алгоритмом ADWIN з використанням помилки класифікації. Як можна помітити з даного зображення, кількість дрейфів, які були визначені алгоритмом KSWIN помітно більше, ніж у розробленого алгоритму з використанням невизначеності.

6. ВИСНОВКИ

Результати експериментів демонструють, що розроблений метод виявлення дрейфів концепцій демонструє меншу кількість спрацьовувань ніж алгоритм KSWIN і на відмінну від алгоритму ADWIN з використанням рівня помилки не вимагає доступу до справжніх міток класів. Але даний алгоритм в поточній реалізації має і ряд недоліків. Найбільшим недоліком даного алгоритму можна назвати те, що для його донавчання після визначення дрейфу концепції він досі вимагає наявності доступу до реальних міток класу. Даний недолік усунути на даний момент неможливо, але можливо зменшити кількість даних для розмічування через введення технік активного навчання. Також серед недоліків даного алгоритму можна визначити більш повільну роботу у порівнянні з алгоритмом KSWIN. Причиною цього слугує те наявність процесу донавчання алгоритму, що при великій кількості нейронних мереж в ансамблі помітно впливає на швидкість роботи, що в свою чергу погано підходить до мереж з великим об'ємом даних.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gulcan EB, Can F (2023) Unsupervised concept drift detection for multi-label data streams. *Artificial Intelligence Review* 56(3):2401–2434
2. Gama, João & Medas, Pedro & Castillo, Gladys & Rodrigues, Pedro. (2004). Learning with Drift Detection. *Intelligent Data Analysis*. 8. 286-295. 10.1007/978-3-540-28645-5_29.
3. Baena-García, Manuel & Campo-Ávila, José & Fidalgo-Merino, Raúl & Bifet, Albert & Gavald, Ricard & Morales-Bueno, Rafael. (2006). Early Drift Detection Method.
4. Bifet, Albert & Gavaldà, Ricard. (2007). Learning from Time-Changing Data with Adaptive Windowing. *Proceedings of the 7th SIAM International Conference on Data Mining*. 7. 10.1137/1.9781611972771.42.
5. Raab, Christoph & Heusinger, Moritz & Schleif, Frank-Michael. (2020). Reactive Soft Prototype Computing for Concept Drift Streams. *Neurocomputing*. 416. 10.1016/j.neucom.2019.11.111.

МЕТОДОЛОГІЇ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ НА ПЛАТФОРМИ iOS

Люлька Р.О.¹, Харченко К.В.²

КПІ ім. Ігоря Сікорського, Кафедра системного проектування, Київ, Україна

¹ lyulka.ruslan@iit.kpi.ua, ² konst1970@gmail.com

У час цифрових інновацій, важливість розробки мобільних додатків для медичних систем особливо зростає. Платформа iOS, відзначаючись безпекою та стабільністю, вимагає специфічних підходів у розробці. Ця робота фокусується на методологіях створення таких додатків, акцентуючи на їх відповідності до високих стандартів медичної галузі на iOS.

Ключові слова: Медичні інформаційні системи, мобільні додатки, методології розробки, Apple Watch.

1. ВСТУП

У сучасному світі, де медицина тісно переплітається з цифровими технологіями, мобільні додатки відіграють ключову роль у підвищенні ефективності медичних послуг та забезпеченні доступності охорони здоров'я. Ці додатки не тільки спрощують процес діагностики та лікування, але й забезпечують надійний засіб комунікації між пацієнтами та медичними працівниками, зокрема в контексті віддаленого моніторингу стану здоров'я та електронного здоров'я.

2. ОСОБЛИВОСТІ ПЛАТФОРМИ iOS ДЛЯ МЕДИЧНИХ ДОДАТКІВ

Платформа iOS від Apple відома своєю високою продуктивністю та надійністю, що робить її ідеальною для розробки медичних додатків. Завдяки оптимізованій архітектурі операційної системи та інтеграції з потужним апаратним забезпеченням [1], iOS дозволяє створювати швидкі та ефективні додатки. Це особливо важливо для медичних застосунків, які потребують швидкої обробки великих обсягів даних та надійності у реальному часі. Також платформа надає розширені можливості для інтеграції з зовнішніми пристроями та сенсорами, що є критично важливим для моніторингу здоров'я та діагностики.

Безпека та конфіденційність даних є основоположними принципами при розробці медичних додатків, і iOS пропонує суворі стандарти та інструменти для їх дотримання. Система безпеки iOS включає шифрування даних на рівні пристрою, захист доступу через Face ID або Touch ID, а також безпечну роботу з хмарними послугами. Це гарантує, що чутливі медичні дані, такі як історія хвороби, лабораторні результати, особисті відомості пацієнтів, захищені від несанкціонованого доступу. Крім того, Apple надає набір рекомендацій та найкращих практик для розробників, щоб забезпечити відповідність додатків законодавчим стандартам, таким як HIPAA в США, щодо обробки та зберігання медичних даних.

3. МЕТОДОЛОГІЇ РОЗРОБКИ

Розвиток мобільних медичних додатків вимагає збалансованого підходу до управління проектами та розробки, щоб забезпечити якість, безпеку та відповідність стандартам. На сьогоднішній день, існує кілька популярних методологій [2], які використовуються у розробці програмного забезпечення.

Гнучка методологія Agile заснована на ітеративному розвитку, де проект розбивається на невеликі частини, що дозволяє команді швидко реагувати на зміни вимог та умов проекту. Agile ідеально підходить для проектів, де важливо швидко адаптуватися до нових вимог або змін у ринкових умовах.

Scrum, як один з підходів Agile, що включає в себе короткі цикли розробки (спринти), регулярні зустрічі команди та постійний зворотній зв'язок. Scrum підходить для проектів з частими оновленнями та активною участю замовника у процесі розробки.

Традиційний підхід Waterfall, де кожна стадія проекту (аналіз вимог, проектування, розробка, тестування, впровадження) виконується послідовно. Ця методологія часто використовується у великих проектах з чітко визначеними вимогами та мінімальними змінами в процесі розвитку.

Адаптація методологій під специфіку медичних додатків.

У контексті медичних додатків, важливо адаптувати ці методології, враховуючи специфіку галузі. Особлива увага приділяється контролю якості, безпеці даних та відповідності медичним стандартам. Наприклад, у методології Agile можна інтегрувати додаткові етапи перевірки безпеки та відповідності стандартам перед кожним релізом. Також важливо забезпечити гнучкість у внесенні змін, що є ключовим у медичному середовищі, де вимоги можуть швидко змінюватися.

4. ІНТЕГРАЦІЯ З МЕДИЧНИМИ ІНФОРМАЦІЙНИМИ СИСТЕМАМИ

Ефективна інтеграція мобільних додатків з медичними інформаційними системами є ключовим аспектом у розробці медичних додатків на платформі iOS. Цей процес включає в себе два основних компоненти.

Технології інтеграції (APIs, обмін даними).

Використання API (Application Programming Interface): API дозволяють мобільним додаткам взаємодіяти з медичними інформаційними системами, обмінюючись даними та функціональністю [3]. Це може включати доступ до медичних записів, обробку запитів на аналізи, оновлення стану пацієнтів тощо.

Обмін даними, при якому важливо забезпечити надійний та безпечний обмін даними між мобільними додатками та медичними системами. Це включає використання стандартів, таких як HL7 або FHIR, для ефективною інтеграції та обміну даними у відповідності до медичних та юридичних норм.

Забезпечення сумісності та безперервності роботи.

Медичні додатки повинні бути сумісними з різними інформаційними системами, що використовуються в медичних установах. Це вимагає гнучкого підходу до розробки та тестування додатків.

А також слід брати до уваги безперервність роботи. Медичні додатки мають забезпечувати стабільну та безперебійну роботу, оскільки вони часто використовуються в критичних ситуаціях. Важливо враховувати аспекти надійності та відмовостійкості при розробці.

5. ІНТЕГРАЦІЯ З APPLE WATCH

Особливо відзначимо внесок Apple Watch у розвиток мобільних медичних додатків на платформі iOS. Завдяки своїм функціям моніторингу здоров'я, таким як відстеження серцевого ритму та рівня кисню в крові, ці розумні годинники стають невід'ємною частиною сучасних медичних додатків [4]. Вони не тільки надають цінні дані для постійного моніторингу стану

здоров'я, але й можуть використовуватися для надання негайної допомоги в критичних ситуаціях.

Інтеграція медичних додатків з Apple Watch відкриває можливості для персоналізації медичного догляду. Використовуючи детальну інформацію, отриману з годинника, медичні додатки можуть надавати індивідуальні рекомендації, базуючись на щоденній активності та фізичних показниках користувача. Це не тільки підвищує ефективність лікування, але й сприяє збору даних для медичних досліджень, що розширює наші знання про здоров'я населення.

Крім того, Apple Watch ідеально інтегруються з екосистемою iOS, що дозволяє медичним додаткам працювати ефективно як на iPhone, так і на годинниках. Така інтеграція не тільки покращує функціональність медичних додатків, але й робить їх більш привабливими для кінцевих користувачів, забезпечуючи їм сучасні засоби для контролю над своїм здоров'ям.

6. ПРИКЛАДИ УСПІШНИХ ПРОЕКТІВ

У цьому розділі розглянемо кілька вдало реалізованих проектів розробки мобільних медичних додатків для платформи iOS. Ці приклади демонструють, як інноваційні підходи та ефективне використання технологій можуть сприяти поліпшенню медичних послуг та підвищенню якості догляду за пацієнтами.

Додаток для віддаленого моніторингу пацієнтів.

Опис. Розробка додатку, який дозволяє медичному персоналу відстежувати стан пацієнтів в реальному часі, збираючи дані з носимих пристроїв.

Висновки. Проект показав важливість інтеграції з різними пристроями та платформами, а також необхідність забезпечення високого рівня безпеки даних.

Мобільний додаток для управління лікарняними записами.

Опис. Створення додатку, що надає лікарям та медсестрам можливість швидкого доступу до медичних записів пацієнтів.

Висновки. Ефективність додатка залежить від його зручності та інтуїтивності використання, що особливо важливо у швидкоплинному медичному середовищі.

Додаток для консультацій з лікарями онлайн.

Опис. Розробка платформи для онлайн-консультацій, яка дозволяє пацієнтам отримувати медичні поради віддалено.

Висновки. Проект наголошує на важливості створення безпечного каналу зв'язку між лікарями та пацієнтами, а також на необхідності інтеграції з іншими медичними системами.

Ці приклади відображають різні аспекти розробки медичних додатків, включаючи технічні виклики, важливість користувацького досвіду, а також необхідність дотримання високих стандартів безпеки та конфіденційності. Вони свідчать про те, що успішні медичні додатки поєднують у собі інноваційні технічні рішення з глибоким розумінням потреб кінцевих користувачів.

7. ВИКЛИКИ ТА ПЕРСПЕКТИВИ

Розвиток мобільних медичних додатків стоїть перед рядом викликів, які вимагають уваги та інноваційних підходів [5]. Водночас, галузь має значний потенціал для розвитку, обіцяючи нові можливості для поліпшення медичного обслуговування.

7.1. Поточні Виклики

1. Забезпечення безпеки та конфіденційності даних. Одним з найбільших викликів є захист чутливої медичної інформації, що обробляється додатками.

2. Інтеграція з існуючими медичними системами. Ефективна сумісність та інтеграція з різними медичними інформаційними системами залишаються складним завданням.

3. Універсальність та адаптивність. Розробка додатків, що враховують різноманіття медичних потреб та пристроїв користувачів.

4. Регуляторні вимоги. Слідкування за змінами в законодавстві та відповідність регулятивним нормам у різних регіонах.

7.2. Майбутні тренди та напрямки розвитку.

Штучний інтелект та машинне навчання. Використання AI для покращення діагностики, персоналізації лікування та аналізу медичних даних.

Інтероперабельність даних, що передбачає розвиток стандартів для поліпшення обміну даними між різними медичними системами.

Телемедицина як розширення можливостей для віддаленого моніторингу та консультацій, особливо в умовах глобальної пандемії.

Персоналізована медицина, що передбачає розробку додатків, що забезпечують індивідуальний підхід до кожного пацієнта, з урахуванням його унікальних потреб та історії здоров'я.

У світлі цих викликів та перспектив, розвиток мобільних медичних додатків на платформі iOS відкриває широкі можливості для інновацій та покращення якості медичного обслуговування на глобальному рівні.

8. ВИСНОВКИ

У світлі стрімкого технологічного прогресу та росту потреб у сфері охорони здоров'я, розвиток мобільних додатків для медичних систем на платформі iOS відіграє ключову роль у модернізації медичного обслуговування. Ці додатки відкривають нові можливості для покращення якості догляду, підвищуючи доступність та ефективність медичних послуг.

Адаптація та впровадження інноваційних технологій у розробці мобільних медичних додатків на iOS перетворюють медичну практику, роблячи її більш інтегрованою та пацієнтоорієнтованою. Майбутнє медичних додатків обіцяє ще більшу інтеграцію з розширеними технологіями, такими як штучний інтелект та телемедицина, що розкриють нові горизонти у лікуванні та моніторингу пацієнтів.

Однак, слід враховувати виклики, пов'язані з безпекою даних, інтеграцією з існуючими системами та регуляторними вимогами. Ефективна реалізація цих додатків вимагає глибокого розуміння медичних потреб, технічної експертизи, та комплексного підходу до розробки, включаючи вибір відповідних методологій та стратегій інтеграції. Таким чином, розвиток мобільних додатків на платформі iOS стане визначальним фактором у формуванні майбутнього медичного обслуговування, оскільки вони пропонують необхідну гнучкість, доступність та персоналізацію у догляді за пацієнтами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. iOS App Development in Healthcare – Features & Cost URL: <https://www.appventurez.com/blog/ios-app-development-in-healthcare> (дата звернення: 26.11.2023).

2. Development Methodologies. URL: <https://www.uptech.team/blog/software-development-methodologies> (дата звернення: 28.11.2023).

3. Zhang, R.; Burgess, E.R.; Reddy, M.C.; Rothrock, N.E.; Bhatt, S.; Rasmussen, L.V.; Butt, Z.; Starren, J.B. Provider perspectives on the integration of patient-reported outcomes in an electronic health record. JAMIA 2019

4. Apple Watch: Creating a Connected Health Ecosystem. URL: <https://www.boston-technology.com/blog/apple-watch-creating-a-connected-health-ecosystem> (дата звернення: 28.11.2023).

5. Mobile healthcare. URL: https://www.researchgate.net/publication/340917395_Mobile_healthcare_opportunities_challenges_prospects (дата звернення: 28.11.2023).

РОЗРОБКА І КОНФІГУРАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ РОЗПОДІЛЕНОЇ СИСТЕМИ КОНТРОЛЮ СПОЖИВАННЯ ЕЛЕКТРОЕНЕРГІЇ В БАГАТОКВАРТИРНОМУ БУДИНКУ

Мунтян Д.М.¹, Кисельов Г.Д.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Інститут прикладного системного аналізу, Київ, Україна

¹ daniilmuntjan@gmail.com, ² g.kyselov@gmail.com

Актуальність теми обумовлена викликами, з якими наша країна зіткнулась останньої зими. Обстріли та енергетична криза виявили потребу в оптимізації споживання енергоресурсів та пошуку ефективних шляхів їх економії. Створення інтелектуальної системи, яка б дозволяла в разі критичних ситуацій оперативно регулювати навантаження та обмежувати споживання електроенергії, не лише сприятиме підвищенню енергетичної безпеки, а й допоможе мінімізувати незручності для громадян. Це вирішить частину проблем, які виникають під час екстремальних ситуацій та може допомогти уникнути нераціональних відключень у багатоквартирних будинках, зокрема, за рахунок надмірного споживання окремими квартирами.

Ключові слова: контроль споживання електроенергії, енергоефективність, оптимізація, енергетична криза, IoT.

1. ВСТУП

В умовах глобальної цифровізації та активного впровадження концепцій розумного будинку (Smart Home), інтеграція інноваційних IoT-рішень у системи управління енергоспоживанням набуває особливої актуальності [1]. Ринок нерухомості швидко реагує на новітні технологічні тренди, намагаючись збільшити комфорт та енергоефективність будівель. Це стає можливим завдяки застосуванню сучасних систем, здатних до автоматизації та гнучкого реагування на зміни в енергетичній інфраструктурі [2]. Розуміння цих процесів та адаптація до них є ключовими для забезпечення стійкості та надійності в управлінні великими житловими комплексами в сучасних умовах.

Непередбачуваність енергопостачання стала реалією сучасної України, особливо в умовах зимових обстрілів, що наклали жорсткі обмеження на використання електроенергії. Зіткнувшись із необхідністю раптового та частково необґрунтованого відключення житлових масивів, виникає питання про розробку більш витончених та ефективних систем управління енергоспоживанням. Ця дипломна робота спрямована на розробку інтелектуальної розподіленої системи, яка дозволить зберегти живлення будинку при обмежених ресурсах генерації електроенергії.

У даній статті концентрується увага на аналізі одного багатоквартирного будинку, проте важливо підкреслити, що розроблене програмне забезпечення має здатність ефективно адаптуватися та масштабуватися для керування енергоспоживанням у масштабах необмеженої кількості будівель. Така гнучкість обумовлена принципами, закладеними в основу даної сервіс-орієнтованої хмарної архітектури.

Основними характеристиками системи є універсальність, абстракція та розширюваність (система може бути легко модифікована чи доповнена новими функціями або компонентами).

Тобто за мету ставиться абстрагуватись від конкретних джерел вхідних даних та зосередитись на програмній частині реалізації, що буде працювати з будь-яким «залізом» та в будь-якому середовищі виконувати поставлені концептуальні задачі. Що конкретно це означає – розберемо у даній статті.

2. КОНТЕКСТ ТА МОТИВАЦІЯ

Зима 2022–2023 років увійшла в історію України як час випробувань, коли доля кинула виклик нашій енергетичній системі. Зіткнувшись із несподіваними масовими відключеннями електроенергії, ми, як суспільство, побачили які вразливі місця має наша інфраструктура. Рисунок 1, який ви можете побачити нижче, слугує свідченням тих перешкод, з якими ми зіткнулися. Відключення електроенергії не тільки призводить до дискомфорту громадян, але й ставить під загрозу функціонування критично важливих структур.

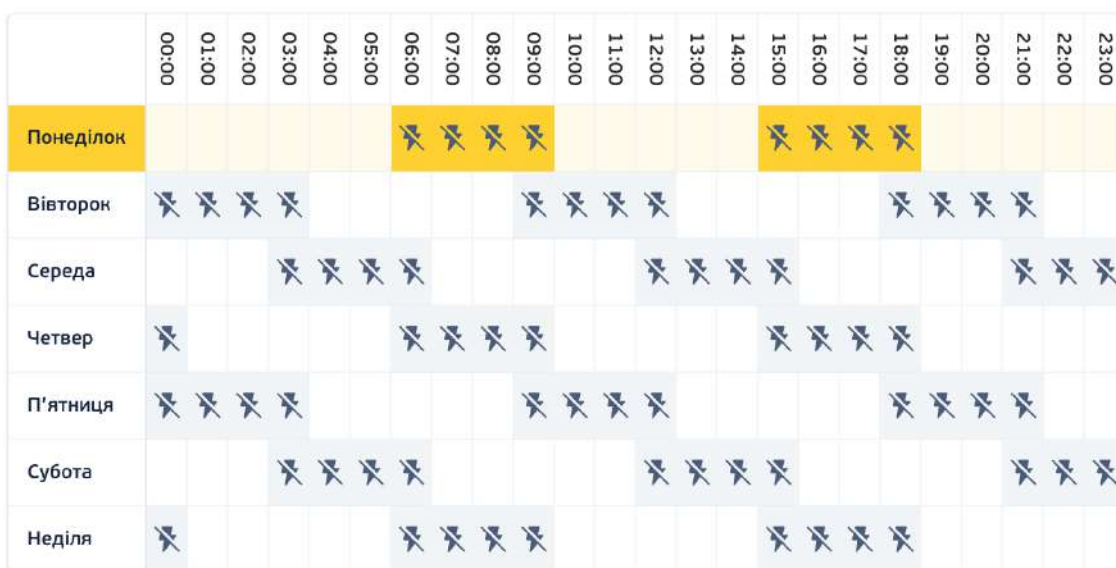


Рисунок 1. Графік відключення світла у Києві

Дефіцит генерації, що впливає з ряду причин, включаючи природні стихії та наслідки воєнних дій, є критичним чинником, який спричиняє згадане перенавантаження системи. Це призводить до каскадних збоїв, що можуть зачіпати цілі райони, залишаючи їх без електропостачання через неможливість вузлів передавати необхідну потужність.

Наразі, електромережі поділяються на категорії надійності. Приватні оселі та комерційні об'єкти належать до третьої категорії приєднання, тоді як друга категорія включає системи, які важливо утримувати в робочому стані, наприклад, вуличне освітлення та світлофори. Найвищий пріоритет надається об'єктам першої категорії, до яких належать поліцейські ділянки, військові об'єкти та медичні установи.

Проте, ця класифікація є надмірно абстрактною та не враховує динаміку індивідуального споживання. Виходом із ситуації може стати більш деталізований підхід, який включає декомпозицію енергетичної системи до рівня окремих квартир. Така деталізація дозволить розподілити навантаження більш раціонально, уникнути загальнобудинкових відключень та забезпечити енергію для тих, хто її потребує найбільше.

Основною мотивацією цієї роботи є створення системи, яка здатна оптимізувати споживання електроенергії на рівні окремого житлового комплексу. Це рішення спрямоване на попередження відключень електроенергії в умовах, коли непомірне споживання невеликої кількості квартир може спричинити неприємності для всього будинку.

3. АРХІТЕКТУРА

На рисунку 2 представлена схема інтелектуальної розподіленої системи контролю споживання електроенергії для багатоквартирного будинку. Ця схема ілюструє зв'язок між окремими компонентами системи та їх взаємодію з централізованим сервером, який виступає як командний центр обробки та управління даними.

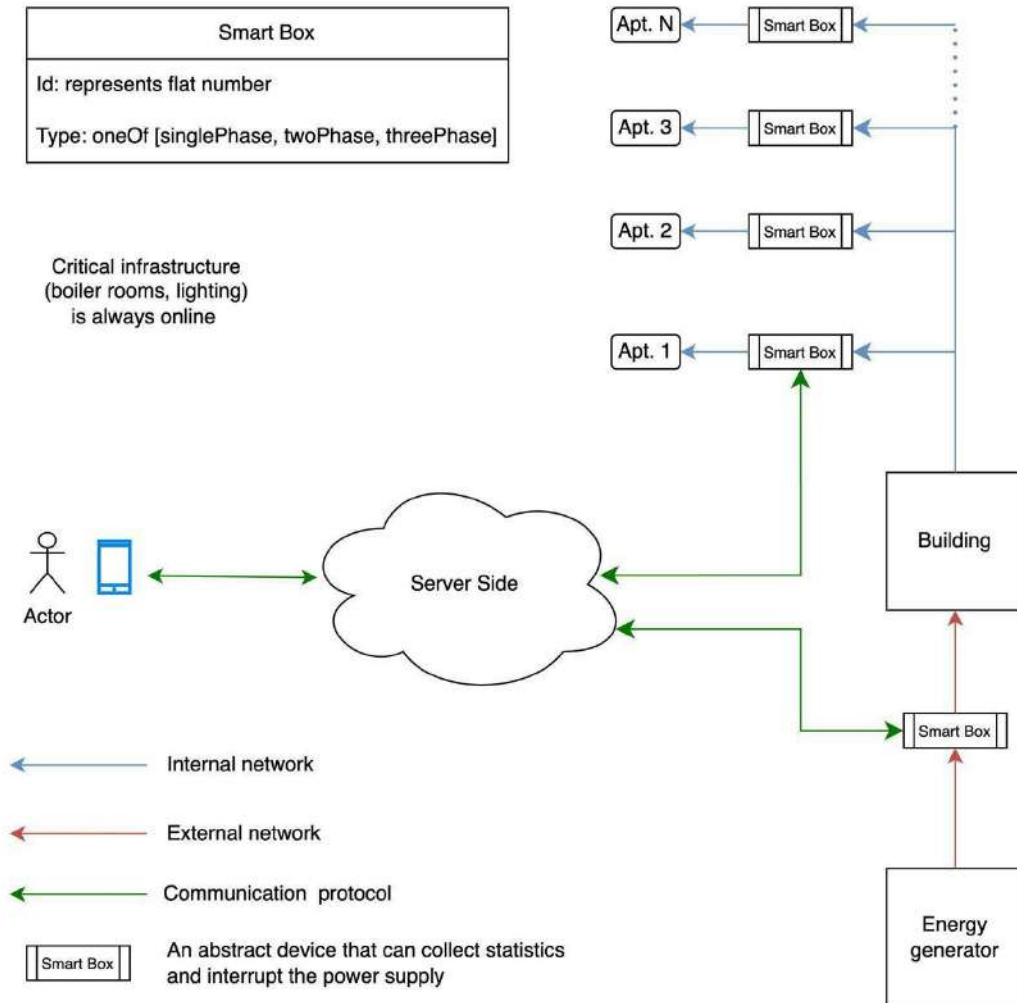


Рисунок 2. Hardware diagram

Розберемо всі сутності, представлені вище.

Smart Vox – це ключовий елемент системи, інноваційний IoT пристрій, що надає змогу вимірювати споживання електроенергії у кожній квартирі та у разі необхідності переривати електропостачання. Кожен Smart Vox має унікальний ідентифікатор, який відповідає номеру квартири, та налаштований на роботу з однофазним, двофазним або трифазним типом електропостачання, забезпечуючи гнучкість та точність у відповідності до специфікацій споживання.

Ядро системи становить мережа цих пристроїв Smart Vox, яка забезпечують зв'язок між індивідуальними квартирами та централізованим сервером, використовуючи головний мікросервер в будинку. Ця складова з'єднується з іншими Smart Vox через GSM-мережу [3], що стандартизує комунікаційні процеси в рамках сучасних IoT-рішень.

Централізована серверна сторона функціонує як «мозок» системи, що здійснює обробку вхідних даних, вироблення команд та диспетчеризацію завдань. Вона аналізує та обробляє інформацію, отриману від кожного Smart Vox [4]. Критично важливі системи, такі як котельні та освітлення, постійно отримують живлення для забезпечення стабільності життєво необхідних функцій.

Користувачі системи, або актори, здебільшого мешканці будинку, отримують інтерактивні сповіщення та можливість керування своїм споживанням через інтуїтивно зрозумілий інтерфейс, доступний через веб або мобільні додатки. Використовуючи ці інструменти, мешканці можуть реагувати на повідомлення про високе споживання в реальному часі, що дозволяє їм приймати необхідні заходи для зниження споживання або підготовки до можливого відключення електроенергії.

Енергогенератор – це джерело генерації електроенергії для будинку. На даному етапі ми розглядаємо звичайний генератор, але система відкрита до інтеграції з різними джерелами енергії.

Зазначена архітектура вирізняється своєю масштабованістю та легкістю інтеграції в існуючі енергетичні системи, що робить її ідеальною не лише для нових «smart houses», але й для модернізації старих будинків [5]. Її універсальність полягає в незалежності від віку чи типу будівлі, надаючи можливість кожному житловому комплексу стати більш енергоефективним та відповідальним у використанні енергоресурсів.

Ця система є прикладом того, як передові технології можуть сприяти створенню більш сталого та енергетично збалансованого суспільства, відповідаючи на виклики енергетичної кризи та високого споживання ресурсів, і в той же час забезпечуючи високий рівень комфорту для своїх користувачів.

4. ХАРАКТЕРИСТИКИ СИСТЕМИ

Основними характеристиками сучасного хмарного продукту є функціональні та нефункціональні вимоги. Їх чітке визначення є надважливим етапом розробки, системи, особливо коли мова йде про сервіс-орієнтовані рішення (service-oriented architecture) [6].

Функціональні вимоги встановлюють конкретні задачі та функції, які програма має виконувати, тоді як нефункціональні вимоги охоплюють очікування до продуктивності, надійності та інтерфейсу користувача. Ретельне розмежування цих вимог є ключем до створення надійної системи, що здатна ефективно вирішувати поставлені завдання, забезпечуючи при цьому високий рівень задоволення користувачів [7].

Основні функціональні вимоги:

- Моніторинг споживання: програмне забезпечення систематично обробляє дані споживання електроенергії в реальному часі, отримані від Smart Vox пристроїв, забезпечуючи точне та актуальне відображення енергетичного профілю кожної квартири.
- Збереження даних: всі зібрані та оброблені дані систематично записуються в базу даних для створення історичного архіву, що використовується для аналітики, звітності та стратегічного планування.
- Віддалене управління: при необхідності, система має можливість дистанційно відключити або підключити електроенергію до окремих квартир або будинку в цілому. Даний функціонал через API інтерфейс надає мережа IoT пристроїв Smart Vox).
- Автоматичне реагування на перевантаження: в разі виявлення або наближення до критичної точки навантаження на генератор, система автоматично вживає заходів для запобігання відключення всього будинку від електромережі, використовуючи для цього систему сповіщення та силового відключення в критичному випадку.

Основні нефункціональні вимоги [8]:

- Доступність (availability): система доступна для користувачів більшу частину часу, мінімізуючи періоди простою.
- Розширюваність (extensibility): архітектура системи дозволяє легке додавання нового функціоналу або інтеграцію з іншими системами без значного перепроектування існуючої структури.
- Спостережливість (observability): ключова характеристика хмарного сервісу — спостережливість, яка дозволяє розробникам та адміністраторам системи відстежувати, аналізувати та розуміти поточний стан системи та її історичну поведінку для швидкого виявлення та вирішення проблем.
- Масштабованість (scalability): система підтримує зростання кількості даних та користувачів без втрати продуктивності.
- Відмовостійкість (fault tolerance): система здатна продовжувати роботу навіть у випадку часткових збоїв, з мінімальним впливом на загальну функціональність.

5. ВИСНОВКИ

Враховуючи сучасні проблеми та потреби у сфері енергетики, можна впевнено стверджувати, що впровадження запропонованої системи може значно вдосконалити ефективність використання електроенергії та сприяти стабілізації енергетичної системи країни, адаптуючи її до непередбачуваних змін та викликів сучасності. Ще на початковій стадії розробки системи було проведено ґрунтовне дослідження, включаючи консультації з електротехнічними спеціалістами та енергетиками, що дозволило забезпечити реалістичність та практичність проекту.

Розроблена система є унікальним та інноваційним рішенням, яке не має прямих аналогів на ринку. Її універсальність дозволяє застосовувати систему до різних типів будинків, без залежності від їх конструктивних особливостей чи віку. Масштабування системи можливе без втрати продуктивності, навіть при збільшенні кількості підключених будинків, за рахунок інноваційних сервіс-орієнтованих підходів.

Інтеграція системи в існуючу енергетичну інфраструктуру не потребує кардинальних змін або додаткових витрат на реконструкцію електричних мереж, оскільки вона розроблена для використання з наявним обладнанням. Це говорить про високу адаптивність та економічну доцільність впровадження системи.

Соціальний аспект реалізації такого рішення є не менш важливим. Оскільки проект має прямий вплив на повсякденне життя мешканців, ключовим є досягнення загальної згоди та встановлення ефективних домовленостей серед них. Важливо, щоб система була використана не лише як технічний інструмент, а й як засіб для зміцнення взаєморозуміння та співпраці між мешканцями. Це вимагає ретельного врахування різних думок та потреб, відкритого діалогу, а також розробки гнучких механізмів для досягнення консенсусу щодо способів управління споживанням електроенергії. Такий підхід не лише сприяє формуванню відповідального ставлення до використання ресурсів, але й підкреслює важливість спільного вирішення проблем, пов'язаних з енергетичними обмеженнями та їх оптимізацією.

Отже, потенціал розвитку системи є значним. У цій роботі як джерело енергії розглядається використання генератора, однак система може бути адаптована до співпраці з великими енергопостачальниками, такими як ДТЕК. У випадках дефіциту генерації енергії, система може отримувати сигнали від постачальників та адекватно реагувати, мінімізуючи навантаження та запобігаючи можливим відключенням.

Таким чином, розроблена система має всі шанси стати важливим інструментом у руках житлово-комунальних організацій, сприяючи сталому розвитку енергетичної сфери та підвищенню якості життя в багатоквартирних будинках.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Carol L. Stimmel та Mark G. Milroy, "IoT and Smart Building Technologies: An Integrated Approach", ISBN: 978-1804619865
2. Keith Moss, "Energy Management in Buildings: The Earthscan Expert Guide", ISBN: 978-0415706469.
3. IoT Journal, "Рішення для розумного будинку", URL: <https://iotji.io/ru/solutions-ru/> (дата звертання: 21.11.2023).
4. Smart МАІС, "Системи розумного будинку", URL: <https://smart-maic.com/> (дата звертання: 21.11.2023).
5. AWS, "Що таке сервіс-орієнтована архітектура", URL: <https://aws.amazon.com/ru/what-is/service-oriented-architecture/> (дата звертання: 22.11.2023).
6. Wikipedia, "Service-Oriented Architecture", URL: <https://w.wiki/8JrA> (дата звертання: 22.11.2023).
7. Thomas Erl, "Service-Oriented Architecture: Concepts, Technology, and Design", ISBN: 978-0131858589.
8. Dan Woods, Thomas Mattern, "Enterprise SOA: Designing IT for Business Innovation", ISBN: 978-0596102388.

СЕРВЕР ДЛЯ МОБІЛЬНИХ ЗАСТОСУНКІВ ЯК СЕРВІС

Науменко Є.О.¹, Булах Б.В.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Інститут прикладного системного аналізу, Київ, Україна

¹ zhenyanaumenko20@gmail.com, ² bogdan.bulakh@gmail.com

Стаття присвячена дослідженню та аналізу Mobile Backend as a Service (MBaaS) як ключового елемента у розробці мобільних додатків. У роботі описуються методи та стратегії використання MBaaS, а також проводиться порівняльний аналіз популярних платформ цього типу. Результати проведеного дослідження дозволили окреслити перелік переваг від використання MBaaS у розробці мобільних додатків порівняно з традиційними підходами.

Ключові слова: сервіс, сервер, мобільний застосунок.

1. ВСТУП

Сучасний світ переживає надзвичайно активний розвиток цифрових технологій, і розробка мобільних застосунків займає важливе місце в цьому процесі. Завдяки зростанню популярності смартфонів і планшетів, а також постійному підвищенню їхньої обчислювальної потужності, мобільні додатки стали неодмінною складовою нашого повсякденного життя. У зв'язку з цим, розробка мобільних застосунків стала не лише технологічно важливою, але і дуже популярною та конкурентною галуззю.

Стаття спрямована на дослідження та розробку мобільного бекенду як сервісу (Mobile Backend As a Service – MBaaS). Ця тема є актуальною через стрімкий розвиток сучасних мобільних додатків. MBaaS відіграє важливу роль у великій кількості мобільних застосунків, надаючи інфраструктуру та функціональність для взаємодії мобільних клієнтів з сервером. Це дозволяє розробникам додатків зосередитися на розробці зовнішньої частини застосунку, забезпечуючи гнучкість та швидкість розробки.

2. ЩО ТАКЕ МВААС?

Mobile Backend as a Service (MBaaS) представляє собою технологічну парадигму, що радикально змінює підхід до розробки мобільних додатків. Це не лише платформа чи набір інструментів – це концепція, яка визнає, що складність бекенд-розробки може бути вирішена через стандартизовані послуги в хмарному середовищі.

MBaaS пропонує готові рішення для складних серверних завдань, які зазвичай вимагають великої кількості часу та експертизи для їхньої реалізації. MBaaS надає розробникам готові модулі та сервіси через інтерфейс програмування додатків (API), що дозволяє значно спростити процес імплементації серверної логіки. Це охоплює створення та управління базами даних, обробку запитів, аутентифікацію, авторизацію користувачів, аналітику, моніторинг, управління контентом, інтеграцію з іншими сервісами та багато іншого.

Цей підхід революціонує спосіб, яким розробники підходять до створення мобільних додатків, дозволяючи їм уникнути важливих технічних складнощів і зосередитися на кращих практиках у фронтенді. MBaaS забезпечує високу гнучкість, масштабованість та безпеку за рахунок використання сучасних технологій хмарних обчислень та безперервного моніторингу.

Це не лише прискорює час від ідеї до запуску продукту на ринок, але й знижує витрати на інфраструктуру та підтримку додатку. Завдяки MBaaS розробники отримують можливість

швидко реагувати на зміни у вимогах ринку та швидше впроваджувати нові функції, забезпечуючи при цьому якість та стабільність продукту на високому рівні.

3. ПОПУЛЯРНІСТЬ ТЕХНОЛОГІЙ

Використання MBaaS набирає популярність у світі розробки мобільних додатків завдяки ряду переваг. Ця платформа дозволяє значно збільшити швидкість розробки та виведення продукту на ринок, зменшити витрати на серверну інфраструктуру та підтримку, а також полегшити управління додатками.

Кілька прикладів компаній, які успішно використовують MBaaS у своїх продуктах:

1. **Uber**: Завдяки використанню MBaaS, Uber зосередився на розробці та вдосконаленні клієнтської частини додатку, в той час як бекендні операції (робота з користувачами, транзакції тощо) були легко та ефективно вирішені через MBaaS-платформи.

2. **Snapchat**: Платформа Snapchat використовує MBaaS для миттєвої обробки великого обсягу медіа-файлів та швидкої доставки повідомлень. Це дозволяє зосередитися на інтерфейсі додатку та функціональності, які пропонують користувачам унікальний досвід.

3. **WhatsApp**: WhatsApp, один із найпопулярніших месенджерів у світі, використовує MBaaS для забезпечення безпеки, аутентифікації та обробки великого потоку повідомлень у реальному часі.

4. **Airbnb**: Популярна платформа для бронювання помешкань використовує MBaaS для обробки операцій з бронюванням, управління користувачами та аналітики даних. Це дозволяє їм зосередитися на розвитку функціональності додатку для кращого взаємодії з користувачами.

5. **Fitbit**: Виробник популярних фітнес-трекерів використовує MBaaS для зберігання та аналізу даних користувачів щодо фізичної активності, серцевого ритму та сну, надаючи унікальні функції для моніторингу здоров'я.

6. **Shazam**: Сервіс розпізнавання музики використовує MBaaS для швидкого обміну даними між користувачами та серверами, що дозволяє миттєво визначати назву та виконавця пісні.

Ці компанії – лише кілька із численних прикладів, що демонструють успішне використання MBaaS у своїх продуктах. Ці приклади ілюструють, як MBaaS дозволяє компаніям сконцентруватися на інноваціях та вдосконаленні користувацького досвіду, мінімізуючи зусилля, витрати та час, необхідні для створення стабільного бекенду.

4. АНАЛІЗ ІСНУЮЧИХ ТЕХНОЛОГІЙ

Перелік популярних технологій

1. Firestore (Google) [1, 2]:

- Переваги: Зручний інтерфейс та швидке створення додатків. Велика спільнота розробників та багато документації. Інтеграція з іншими Google-сервісами, такими як Google Analytics.

- Недоліки: Залежність від інфраструктури Google, що може створювати обмеження для певних користувачів. Обмежена гнучкість у налаштуванні.

2. AWS Amplify (Amazon Web Services) [3]:

- Переваги: Велика масштабованість та гнучкість завдяки інфраструктурі AWS. Широкий вибір підтримуваних програмних мов та фреймворків. Потужність та можливості для великих проєктів.

- Недоліки: Складніший для розуміння для початківців. Вартість може бути високою для малих проєктів.

3. Heroku (Salesforce) [4]:

- Переваги: Простий у використанні інтерфейс. Масштабованість, автоматичне розгортання та кешування. Підтримка різних програмних мов.

- Недоліки: Вартість може бути високою з ростом обсягу. Може бути обмеженіше для специфічних завдань.

4. Kinvey (Progress) [5, 6]:

- Переваги: Простий для початківців та швидкий у розробці. Забезпечує інструменти для мобільних та веб-додатків. Гнучкість у створенні власних бекенд-серверів.

- Недоліки: Може бути менше функціональністю порівняно з іншими платформами. Залежність від одного постачальника.

5. Backendless [7]:

- Переваги: Простий у використанні та гнучкий. Підтримує багато мов програмування. Відкритий вихідний код для більшого контролю.

- Недоліки: Може бути менше популярним порівняно з іншими платформами. Відсутність інтеграції з іншими сервісами.

Вартісне порівняння наведено в табл. 1.

Таблиця 1. Вартісне порівняння

МВaaS-платформа	Найменший тариф (на місяць)	Квота	Додаткова інформація
Firebase	Безкоштовно	10 ГБ бази даних, 5 ГБ хостингу	Обмеження на запити
AWS Amplify	Безкоштовно	1 ГБ зберігання, 15 ГБ трафіку	Включає AWS безпеку
Heroku	Безкоштовно	10 тис. рядків бази даних	Гнучка система
Kinvey	\$200	Необмежена кількість користувачів	Можливі додаткові витрати
Backendless	\$25	1 ГБ бази даних, 50 ГБ трафіку	Підтримка REST і GraphQL

Ці тарифи представляють найдешевші доступні плани для кожної МВaaS-платформи.

На основі порівняння цін за найдешевший тариф, можна зробити наступні висновки:

Firestore: Безкоштовний тариф Firestore доступний, і він має значні включення, такі як 10 ГБ бази даних і 5 ГБ хостингу. Це може бути ідеальним вибором для початкових проектів та тестування [2].

AWS Amplify: також пропонує безкоштовний тариф з 1 ГБ зберігання та 15 ГБ трафіку. Плюсом є доступ до широкого спектру AWS-сервісів для розширення функціональності [3].

Heroku: надає безкоштовний план з обмеженнями, але це дозволяє почати розробку без витрат. Велика перевага полягає в гнучкості системи та підтримці багатьох мов програмування [4].

Kinvey: пропонує високий функціональний тариф за \$200, але це може бути занадто дорого для невеликих проектів [5].

Backendless: надає бюджетний план за \$25, і це може бути привабливим варіантом для малих проектів та стартапів [7].

Завдяки безкоштовному плану Backendless, ця платформа може бути привабливою для невеликих проектів і розробників, які шукають бюджетний варіант. Однак, обираючи МВaaS-платформу, слід також враховувати інші фактори, такі як функціональність, підтримка, доступність SDK і інші параметри [7].

5. ЗАПРОПОНОВАНА СХЕМА РЕАЛІЗАЦІЇ ПОСЛУГИ

Клієнтські додатки, такі як мобільні додатки або веб-додатки, взаємодіють з MBaaS через API для доступу до даних та сервісів.

API Gateway: Шлюз API використовується для маршрутизації запитів від клієнтів до відповідних служб MBaaS. Він також відповідає за аутентифікацію та авторизацію користувачів.

Серверна логіка: Цей компонент містить логіку обробки запитів, обробки даних та виконання бізнес-логіки. Він включає в себе обробку запитів REST або GraphQL, обробку зображень, обробку повідомлень тощо.

База даних: Це сховище для зберігання даних, таких як користувачі, сесії, зображення, повідомлення тощо. База даних може бути NoSQL (наприклад, MongoDB) або SQL (наприклад, PostgreSQL), в залежності від потреб проекту.

Автентифікація та авторизація: Цей компонент відповідає за ідентифікацію користувачів та надання їм доступу до відповідних ресурсів. Він може включати в себе реалізацію протоколів автентифікації, таких як OAuth або OpenID Connect.

Хмарні послуги: Деякі MBaaS можуть інтегрувати хмарні послуги, такі як сховище файлів, Push-сповіщення, інструменти аналітики та багато інших.

Спостереження та моніторинг: Для підтримки та відстеження стану MBaaS, важливо включити засоби спостереження та моніторингу, які допомагають виявляти проблеми та оптимізувати роботу системи.

Скальованість та висока доступність: MBaaS повинен бути розроблений з урахуванням можливості масштабування та забезпечення високої доступності. Це може включати в себе використання контейнерів та систем управління скальованістю.

Зобразимо приблизну схему (приклад) архітектури для власного MBaaS, який використовує Spring Boot, gRPC, Spring Security, MongoDB, push-повідомлення, автентифікацію та авторизацію (рис. 1):

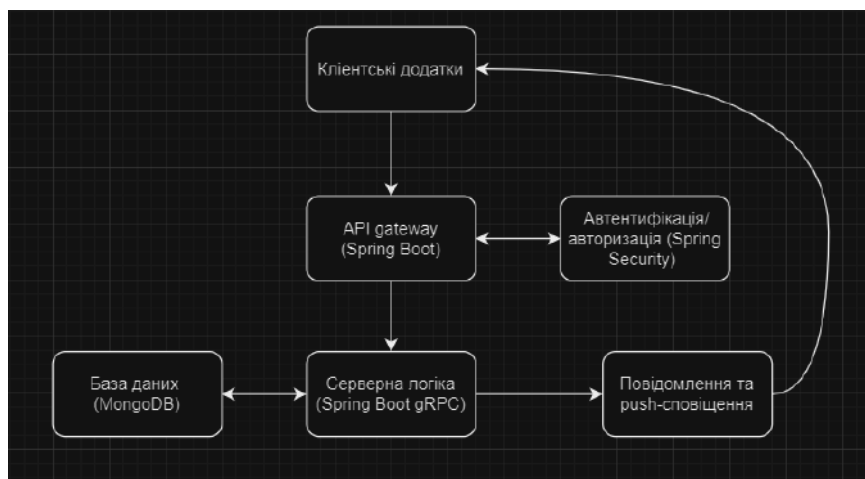


Рисунок 1. Приклад архітектури

У цій схемі:

Клієнтські додатки: мобільні додатки, які взаємодіють з MBaaS через API Gateway.

API Gateway (Spring Boot): Відповідає за маршрутизацію запитів від клієнтів до відповідних сервісів та служб MBaaS. Він також відповідає за забезпечення безпеки через Spring Security.

Spring Security (Автентифікація та авторизація): Використовується для ідентифікації та авторизації користувачів. Встановлює права доступу до ресурсів.

Серверна логіка (Spring Boot, gRPC): Включає в себе обробку запитів та виконання бізнес-логіки. Ви можете використовувати gRPC для взаємодії між сервісами.

База даних (MongoDB): Використовується для зберігання даних, таких як користувачі, сесії та інші. MongoDB відмінно підходить для зберігання документів.

Повідомлення та push-сповіщення: Відповідають за надсилання повідомлень користувачам та реалізацію push-сповіщень для клієнтських додатків.

Ця схема показує загальний підхід до архітектури вашого власного MBaaS.

6. ВИСНОВКИ

У цьому дослідженні було проведено аналіз існуючих підходів до розробки серверної частини мобільних додатків, зокрема, можливостей Mobile Backend as a Service (MBaaS). Було проведено вивчення і порівняння різноманітних MBaaS-платформ, вибір оптимального рішення для конкретного проекту, аналіз можливостей створення власного MBaaS, а також оцінку архітектури та переваг використання MBaaS.

Результати проведеного аналізу показали, що кожна з існуючих MBaaS-платформ, таких як Firebase, AWS Amplify, Heroku, Kinvey та Backendless, має свої переваги та обмеження. Вибір конкретної платформи має базуватися на унікальних потребах та характеристиках конкретного проекту. Firebase виділяється своєю простотою використання та швидкістю розробки, тоді як AWS Amplify надає масштабованість та гнучкість для великих проектів. Heroku дозволяє розгорнути додатки без необхідності управління інфраструктурою, а Kinvey та Backendless надають більше контролю над функціональністю.

В результаті проведених досліджень окреслено переваги та недоліки існуючих хмарних рішень і визначені важливі аспекти щодо вибору конкретного рішення або розробки власного MBaaS для конкретного проекту. Отримані результати є корисними для розробників сучасних мобільних додатків.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Firebase”. URL: <https://firebase.google.com> (дата звернення 23.11.2023)
2. “Firebase documentation”. URL: <https://firebase.google.com/docs> (дата звернення 25.11.2023)
3. “AWS Amplify”. URL: <https://aws.amazon.com/amplify/> (дата звернення 25.11.2023)
4. “Heroku”. URL: <https://www.heroku.com/salesforce> (дата звернення 25.11.2023)
5. “Kinvey (Progress)”. URL: <https://www.heroku.com/> (дата звернення 25.11.2023)
6. “React.js”, Official Documentation. URL: <https://www.progress.com/kinvey> (дата звернення 23.11.2023)
7. “Backendless”. URL: <https://backendless.com/> (дата звернення 25.11.2023)

МЕТОДИ АНАЛІЗУ НОВИН

Орловський А.В.¹, Кислий Р.В.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

¹ artem.orlovsky@gmail.com, ² kvrware@gmail.com

Зі збільшенням і складністю потоку нової інформації виникає потреба у аналізі новин, постановці пріоритетності інформації, пошуку першоджерел та виявлення неправдивих новин чи маніпуляцій. В даній роботі розглядаються методи аналізу новин українського інформаційного простору за допомогою лінгвістичних та нейромережових інструментів обробки природньої мови.

Ключові слова: новина, статистична модель, нейромережева модель, трансформер, векторне представлення

1. ВСТУП

Новини є оперативними інформаційним повідомленнями, які в себе включають важливу інформацію для соціуму. Новини є особливим джерелом інформації для людей в тому числа для прийняття рішень. Для аналізу новин використовуються методи журналістики, а також методи машинного навчання, що здатні автоматизувати обробку даних та виділяти головне з повідомлення, знаходити зв'язки між різними новинами. Для таких задач використовуються методи обробки природньої мови, що складаються як і з класичних методів обробки враховуючи її граматику і лінгвістичні правила, так і методи з використанням мереж глибокого навчання, включаючи нейромережі типу трансформер з механізмом уваги.

2. ЗАДАЧА АНАЛІЗУ НОВИН ЗА ЗМІСТОМ

Для аналізу новин та пошуку першоджерел потрібно розуміти, яким чином можна порівняти новини, виміряти їх близькість одна до одної. Це завдання вимагає математичного підходу та використання методів обробки природньої мови. Основним викликом є перетворення текстової інформації на певне векторне представлення, в просторі якого можна ввести метрики для порівняння схожості чи відмінності різних новин.

Дана математична задача полягає в репрезентації тексту у вигляді числового вектору. У векторному просторі кожен текст стає точкою зі своїм унікальним векторним представленням. Це векторне представлення може бути створене за допомогою різних методів, таких як Bag of Words (BoW) [1], TF-IDF (Term Frequency-Inverse Document Frequency) [2], або більш складні моделі, такі як агрегації Word2Vec [3] або векторні представлення BERT [4].

3. TF-IDF РЕПРЕЗЕНТАЦІЯ ТЕКСТУ

Дана ідея заснована на концепції розподілу слів у текстах. Слова неоднорідні за частотністю в тексті, ймовірність виявити одне слово в текстах може бути більшою за інші. Закон Ципфа – це лінгвостатистичний закон, який стверджує що відношення рангу слова в частотному словнику до частотності слова є константою, тобто чим більше ранг, тим обернено менше частотність, що ілюструється на наступному графіку нижче:

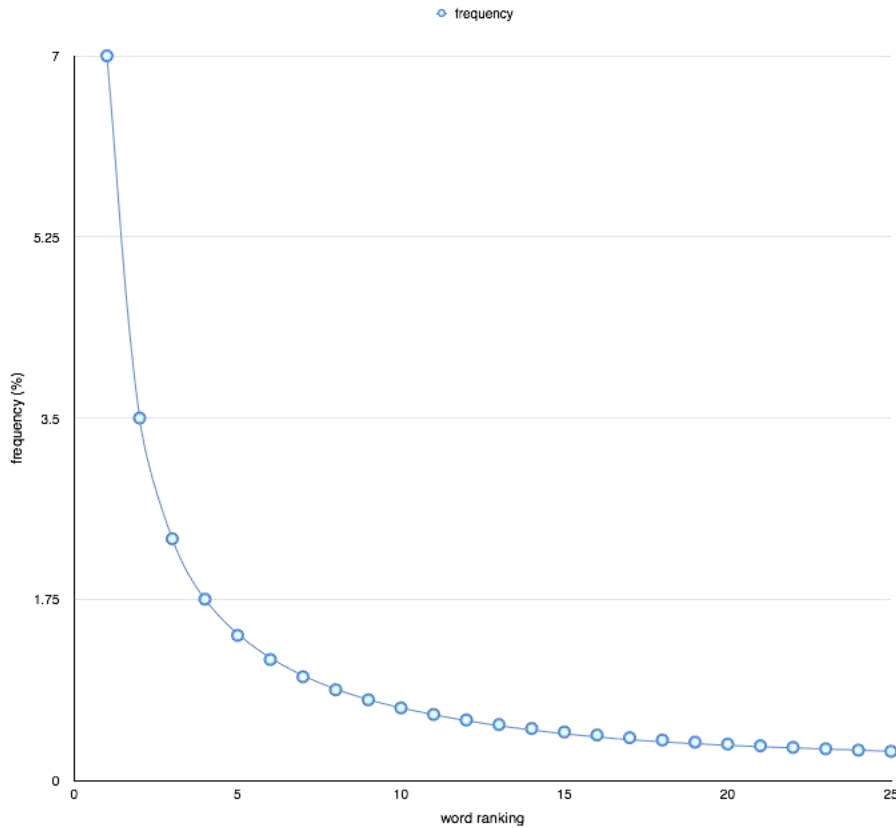


Рисунок 1. Графік розподілу Ципфа, з ранжуванням частотності слів

Враховуючи різну тематику документів, різні слова можуть в одних текстах зустрічатись частіше ніж інші. Загальні слова, що існують у мові, будуть мати велику частоту використання в усіх документах, але деякі слова будуть властиві саме певному документу. Таким чином, можна ввести формулу для прорахунку виміру належності слова конкретному документу в конкретному корпусі слів, і дана формула називається TF-IDF (3) – term frequency (1), inverse document frequency (2).

$$TF(t, d) = \frac{n_t}{\sum_k n_k} \quad (1)$$

$$IDF(t, D) = \log \frac{|D|}{|\{d_i \in D | t \in d_i\}|} \quad (2)$$

$$TF - IDF(t, d, D) = \frac{n_t}{\sum_k n_k} \log \frac{|D|}{|\{d_i \in D | t \in d_i\}|} \quad (3)$$

де n_t – частота слова t в документі d , D - корпус текстів, датасет, $|D|$ -кількість документів в корпусі, $|\{d_i \in D | t \in d_i\}|$ - кількість документів в колекції D де зустрічається слово t .

Якщо обрати словник слів, кількість слів у словнику нехай буде M , то отримаємо векторну репрезентацію документу з довжиною вектора M . Даний вектор не включає в себе виявлення граматичних ознак в тексті чи взаємозв'язність слів для отримання нових сенсів у документі, але на практиці така репрезентація дає гарний бейзлайн у поєднанні з моделлю логістичної регресії (4) для задач класифікації, в тому числі для виявлення пропаганди у тексті, що є однією з важливих задач аналізу новин.

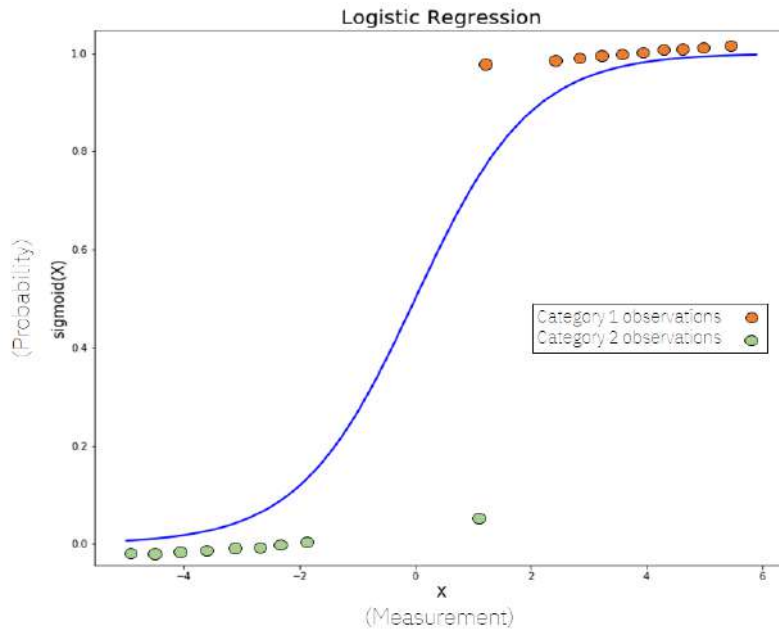


Рисунок 2. Графік логістичної регресії, на графіку зображена крива сігмоїди, для якої обрані параметри під задачу

$$\log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X \quad (4)$$

Формула логістичної регресії (4), де $\log\frac{P}{1-P}$ – лог-шанси, $\beta_0 + \beta_1 X$ – пряма регресії.

4. ВЕКТОРНЕ ПРЕДСТАВЛЕННЯ ТЕКСТУ З ОЗНАК SENTENCE-TRANSFORMER

Один з найефективніших способів отримати векторне представлення текстів за змістом у сучасних моделях високої якості – це використання вихідних векторів з архітектури sentence transformer [5]. Ця інноваційна архітектура заснована на принципі роботи сіамських нейронних мереж та є ефективною для задач обробки природної мови.

Основна ідея полягає в тому, що під час тренування сіамської мережі використовуються пари текстів, і мережа навчається порівнювати їх схожість. При цьому вихід з трансформера проходить через пулінг шар нейронної мережі, а потім відбувається класифікація схожості векторів, часто використовуючи косинусну схожість або додатковий шар з класифікатором Softmax активації. Цей процес навчання дозволяє мережі вчитися представляти текст у векторному просторі таким чином, що схожі текстові документи будуть мати близькі векторні представлення.

Після завершення тренування сіамська мережа може бути використана для отримання векторних представлень текстів. Для цього векторне представлення кожного тексту отримується шляхом відкидання крайнього шару сіамської мережі і обирається лише одна ланка такої мережі.

Використання sentence transformer дозволяє отримувати векторні представлення текстів, які можна використовувати для подальшого аналізу, кластеризації, класифікації та пошуку схожих текстів.

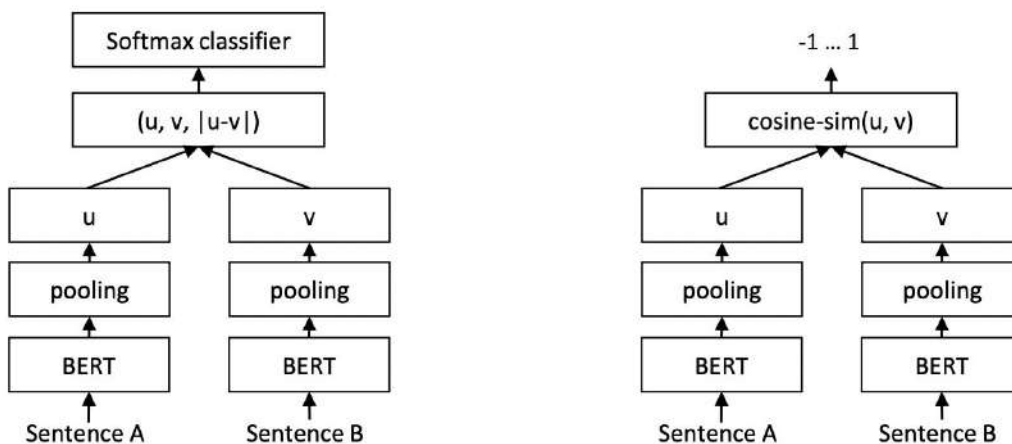


Рисунок 3. Схема архітектури сіамських нейронних мереж sentence-transformers

В основі sentence-transformers лежить архітектура трансформеру BERT, який є нейромережею-енкодером, тобто такою, що вивчає загальний контекст тексту та тренується з двонаправленим навчанням на тексті. Така методика дозволяє отримати контекст для слова, що вивчається, не тільки з попередніх слів, але й з наступних, вивчаючи додаткові взаємозв'язки.

5. ПОШУК СХОЖИХ НОВИН ЗА АЛГОРИТМОМ ПОШУКУ НАЙБЛИЖЧИХ СУСІДІВ

Після отримання функції перетворення текстового документу в певне векторне представлення, у вигляді нейромережі чи класичного статистичного підходу, наступним необхідним кроком для аналізу є створення індексу в отриманій векторній даних та використання належного алгоритму пошуку найближчих сусідів точки у просторі, з яких потім дослідник отримає найближчі новини до тої, що він вивчає, за змістом. З класичних статистичних алгоритмів є алгоритм пошуку k -найближчих сусідів, але він потребує значного часу пошуку за великих даних. У разі аналізу новин важлива швидкість, оскільки для вчасного реагування на дезінформацію її треба виявити якнайскоріше ще коли інформація є актуальною.

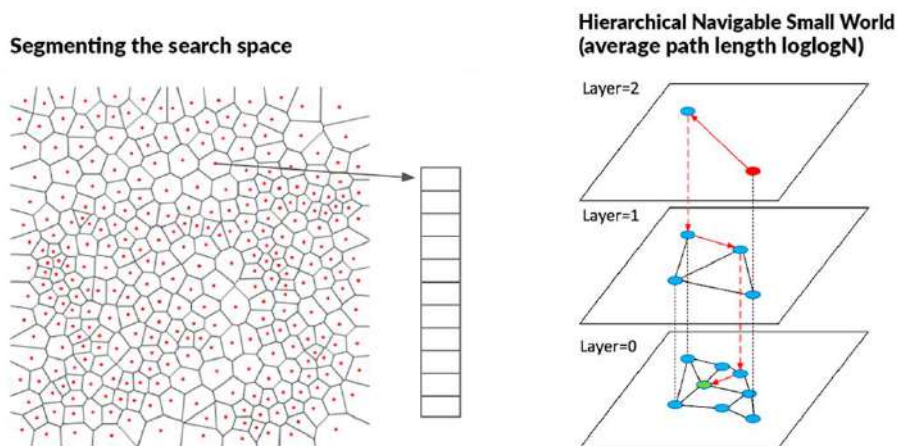


Рисунок 4. Візуалізація основних компонентів алгоритму Hierarchical Navigable Small World (HNSW)

З розробок нових структур даних, що використовуються в сучасних базах даних, застосовуються методи пошуку приблизних найближчих сусідів. Дані структури даних є ймовірнісними, вони можуть не повернути всіх точних сусідів, але швидкість таких алгоритмів в рази більша. В порівнянні, k -найближчих сусідів має оцінку швидкості пошуку $O(n^2)$, в той час як сучасний алгоритм Annoy, враховуючи передрозрахунок структури даних перед пошуком, має складність $O(\log(n))$, що суттєво зменшує час на пошук найближчого сусіда. Складність алгоритму пошуку Hierarchical Navigable Small World, візуалізація якого зображена вище, також має складність $O(\log(n))$, але іншу структуру даних.

Таким чином, даний спосіб індексації дозволить створити апарат швидкого пошуку близьких точок у просторі, а отже для обраної задачі – близькі новин за змістом. Для подальшого аналізу можливо виявляти суперечності між оригінальною новиною та новиною, що на неї послалася, визначати частину тексту яка суперечить чи несхожа за змістом до першої новини, та складати лог особливих подій, які варті для більш глибокого дослідження людиною-експертом.

6. ВИСНОВКИ

У світі з постійним розвитком інформаційних технологій, роль новин як особливого джерела інформації для суспільства набуває особливої актуальності. Новини є оперативним та важливим джерелом інформації, на основі якого можуть робитись рішення в режимі реального часу. Саме тому необхідно створити додаткові інструменти, що могли б спростити проведення аналізу новин людиною.

Постановка даної задачі починається з репрезентації тексту у вигляді вектору чисел різними моделями, такими як Bag of Words, TF-IDF, Word2Vec, BERT, і sentence-transformers. Однаково важливі як і нейромережеві підходи, так і статистичні, оскільки вони більш пояснювані через визначення їх формул та основних засад з теорії ймовірності.

Для подальшого аналізу використовуються структури даних що є основою векторних баз даних, на основі яких відбувається швидкий пошук та аналіз векторного простору певних записів у базі. Отримана векторна форма тексту слугує як і для задачі пошуку найближчих новин за змістом, так і перевірки протиріччя новини за змістом чи пошуку кластеру чи топіку новин за тематикою цієї новини. Дані інструменти будуть спеціальним засобом роботи з базою новин для людей-експертів, що спеціалізуються на аналізі новин.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zellig S. Harris, "Distributional Structure", Word Journal, 1954, Vol. 10, No. 2-3, pp 146-162
2. Sparck Jones, K., "A statistical interpretation of term specificity and its application in retrieval", Journal of Documentation, 1972, Vol. 28 No. 1, pp. 11-21.
3. "Distributed Representations of Words and Phrases and their Compositionality", Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, 2013, URL – <https://arxiv.org/abs/1310.4546>, дата звернення – 27.11.2023
4. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018, URL – <https://arxiv.org/pdf/1810.04805.pdf>, дата звернення – 27.11.2023
5. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", Nils Reimers, Iryna Gurevych, 2019, URL – <https://arxiv.org/abs/1908.10084>, дата звернення – 27.11.2023

ОГЛЯД ПРІОРИТЕЗАЦІЇ ТРАФІКУ В ПРОГРАМНО ВИЗНАЧЕНИХ МЕРЕЖАХ

Переяславський С.К.¹, Письменний І.О.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Інститут прикладного системного аналізу, Київ, Україна

¹ pereiaslavskyi.serhii@lil.kpi.ua, ² ihor.pismenny@gmail.com

З ростом складності розподілених комп'ютерних систем, збільшення їх гетерогенності, актуальності набула ідея відділення контролю трафіку в мережах від фізичних пристроїв - перенесення функції маршрутизації та пріоритизації трафіку в програмну площину. Дана робота присвячена питанню пріоритизації трафіку в гетерогенних розподілених мережах з допомогою програмно-визначених систем (англ. Software-Defined Networking, SDN).

Ключові слова: програмно-визначені мережі, контролер, пріоритизація, трафік.

1. ВСТУП

Програмно визначені мережі (Software Defined Networks) – це інноваційний підхід до побудови мережевої архітектури, який виник у відповідь на сучасні виклики та потреби у галузі інформаційних технологій. SDN може вирішувати цільові задачі завдяки своїй здатності забезпечити більшу гнучкість, ефективність, безпеку та контроль у мережевій інфраструктурі, що стає критично важливим у зв'язку із ростом цифровізації та змінами в бізнес-середовищі і бізнес-процесах.

2. ОСОБЛИВОСТІ SDN

Технологія програмно-визначених мереж є ключовим елементом еволюції мережевих структур у світі технологій, які швидко розвиваються. Програмно визначена мережа означає не тільки те, що мережеві елементи управляються програмно і можуть швидко та ефективно перебудовуватися, але й те, що на одному фізичному пулі мережевих елементів можуть бути розгорнуті багато мереж, що логічно не залежать один від одного. Такі логічні мережі можуть передавати потоки трафіку різних бізнес-додатків, не заважаючи одна на одну.

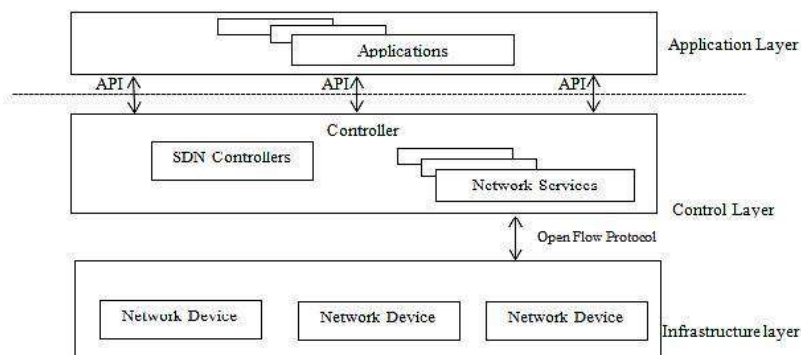


Рисунок 1. Структура SDN

Програми, яким потрібні різні параметри та конфігурації мережі, адмініструють SDN-контролер за допомогою інтерфейсів програмування API (Application Programming Interface).

У такій архітектурі кожен бізнес-додаток може через SDN-контролер сконфігурувати собі логічну мережу із загальних ресурсів мережевої інфраструктури рівня передачі. І кожна така мережа працюватиме незалежно від інших логічних мереж у тому ж пулі ресурсів (за їх достатністю та наявністю).

Для керування інфраструктурою мережі від SDN-контролера необхідний протокол відкритого керування. Такий протокол, розроблений вченими зі Стенфорда, називають OpenFlow (не єдиний можливий протокол для застосування). Робота OpenFlow полягає у модифікації змісту таблиці передачі пакетів усередині маршрутизатора чи комутатора (Forwarding table). Архітектура мережевих пристроїв часто представляється у вигляді трьох площин: адміністрування (management plane), управління (control plane) та передачі даних (data plane).

Традиційно мережеве обладнання від різних виробників часто має спеціальні системи конфігурації та керування, що обмежує взаємодію між маршрутизаторами та комутаторами від різних виробників. Щоб вирішити цю проблему, було створено OpenFlow як відкритий програмований мережевий протокол для налаштування та керування мережевими комутаторами від різних постачальників. Це дає змогу перенести площину керування всіма комутаторами на центральний контролер і дозволяє центральному програмному забезпеченню визначати поведінку мережі. Таким чином, мережеві адміністратори можуть використовувати програмне забезпечення OpenFlow для управління та контролю потоку трафіку між комутаційним обладнанням різних марок.

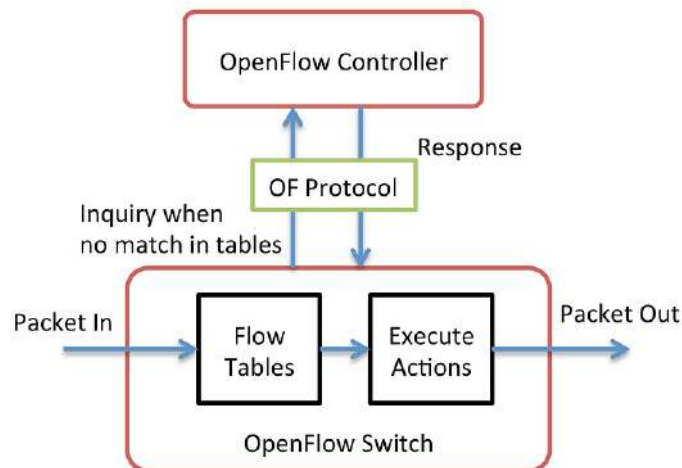


Рисунок 2. Структура OpenFlow комутатора

OpenFlow зазвичай складається з трьох компонентів: контролера, протоколу OpenFlow і ланцюжка таблиць пакетів, налаштованих на комутаторі OpenFlow. Контролер може встановлювати правила щодо поведінки пересилання даних кожного пристрою через протокол OpenFlow. Коли пакет даних надходить до комутатора OpenFlow, комутатор шукатиме відповідні записи пакета в таблицях і виконає відповідні дії. Якщо відповідності не знайдено, подія запиту надсилається до контролера OpenFlow, який потім відповідає новим записом потоку для обробки цього пакета в черзі.

3. ВІДМІННІСТЬ ВІД ТРАДИЦІЙНИХ МЕРЕЖ

На відміну від традиційної архітектури маршрутизатора або комутатора, яка об'єднує функції управління і передачі даних на одному пристрої, SDN пропонує метод віртуалізації мережі [1], який дозволяє програмно впроваджувати різні логічні функції в різних компонентах мережі з метою гнучкого та ефективного керування. Розділення рівня керування

і рівня даних дозволяє маршрутизаторам і комутаторам використовувати дані шару управління для спрямування вхідного трафіку до відповідного інтерфейсу виходу.

Інфраструктура є основною відмінністю між SDN і традиційними мережами. Оскільки рівень керування ґрунтується на програмному забезпеченні, SDN є набагато гнучкішим, ніж традиційна мережа. Це дозволяє адміністраторам контролювати мережу, змінювати параметри конфігурації, надавати ресурси та збільшувати пропускну здатність мережі – все з централізованого інтерфейсу користувача без необхідності встановлення додаткового обладнання.

Традиційні мережі зазвичай мають об'єднану архітектуру, де управління та передача даних зосереджені на окремих фізичних пристроях (маршрутизатори/комутатори). Це призводить до того, що зміни в мережевій конфігурації можуть бути важкими та тривалими завданнями, оскільки вони вимагають фізичної конфігурації кожного пристрою окремо. Натомість мережі SDN відрізняються єдиним підходом до управління мережею. Програмний контролер в цих мережах відокремлений від фізичної інфраструктури і приймає на себе велику частину функцій управління.

В цілому, мережі SDN надають більше гнучкості, контролю, гарантованої безпеки та можливостей моніторингу порівняно з традиційними мережами. Це робить їх особливо корисними для сучасних сценаріїв, де вимоги до мережі зростають разом з розширенням цифрової трансформації та використанням нових технологій [2].

Крім того, існують відмінності в безпеці між SDN і традиційними мережами. SDN забезпечує більшу безпеку в багатьох аспектах завдяки підвищеній видимості та здатності визначати безпечні шляхи. Однак, оскільки програмно визначені мережі покладаються на централізований контролер, захист контролера має важливе значення для підтримки безпеки мережі [3].

У порівнянні з традиційною мережевою архітектурою, централізована контрольна точка архітектури SDN дозволяє ефективно реагувати на проблеми безпеки в мережі. Незалежно від розміру мережі, обсягу даних, що обробляються в мережі тощо, зосередження на безпеці стає дуже важливим предметом. Можливості мережі можна оцінити залежно від її споживання ресурсів, масштабованості, доступності та стійкості до збоїв. Однак усі ці вимірювання будуть безглуздими, якщо мережа буде вразлива до атак. Це підкреслює важливість дослідження поведінки користувачів і недоліків безпеки всередині архітектури SDN.

4. ПРІОРИТЕЗАЦІЯ ТРАФІКУ

У світі великих потоків даних та безперервних комунікацій, вибір правильного маршруту для кожного пакета інформації стає викликом для сучасних мереж. Зі стрімким збільшенням обсягів даних, виникає питання, як оптимально розподілити багатошаровий потік, де дані великого обсягу конкурують за обмежені ресурси. Пріоритетизація трафіку стає невід'ємним підходом, який розподіляє ресурси для забезпечення оптимальної продуктивності мережі.

З різноманіттям послуг та додатків виникає завдання створити гнучку систему, де кожен тип трафіку отримає необхідний рівень обслуговування. Це вимагає докладної конфігурації, щоб уникнути конфліктів та забезпечити ефективне використання ресурсів.

Quality of Service (QoS) визначається як набір механізмів і технологій, призначених для забезпечення певного рівня якості обслуговування в мережевому середовищі. Основна мета QoS полягає в тому, щоб ефективно управляти трафіком, гарантувати передачу даних з певними параметрами та враховувати потреби різних видів додатків та користувачів.

Важливим аспектом управління мережею для забезпечення високої доступності та ефективності є класифікація трафіку. Існують методи застосування класифікації трафіку в

мережах: використання портів для визначення прикладних програм і протоколів прикладного рівня або використання детальної перевірки пакетів (deep packet inspection). Ці методи мають високу точність, однак є деякі проблеми щодо їх реалізації, оскільки динамічні порти та зашифрований трафік ще не підтримуються в поточних мережах. Це також спричиняє великі витрати на систему та порушує конфіденційність користувачів.

Розставляючи пріоритет трафіку на основі попередньо визначених правил, SDN може гарантувати послідовну взаємодію користувача з основними службами, одночасно запобігаючи перевантаженням мережі, спричиненим некритичним трафіком.

5. ПРОПОНОВАНА РЕАЛІЗАЦІЯ

В якості дослідження проблематики пріоритезації трафіку в програмно визначеній мережі, пропонується метод класифікації трафіку для його пріоритезації на основі кращого підходу до навчання моделі-класифікатора [4].

Пропонована реалізація включає в себе проведення підготовчого етапу для формування передумов для подальших досліджень, а саме – для пріоритезації трафіку на основі попередньо досліджених особливостей пакетів.

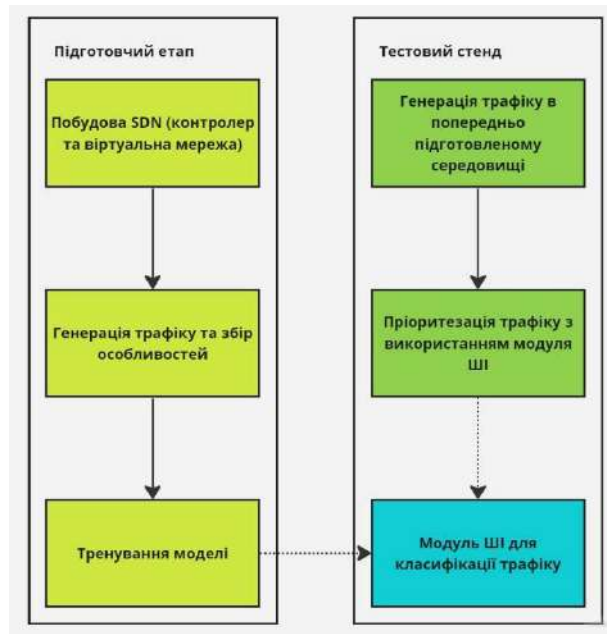


Рисунок 3. Етапи досліджень пропонованої реалізації

При підготовчому етапі передбачається побудова SDN системи разом з контролером та віртуальною мережею, генерація і збір трафіку та подальше застосування його особливостей для тренування класифікатора. Використання тестового стенду передбачає задіяння натренованої моделі-класифікатора для пріоритезації трафіку в режимі реального часу.

6. ВИСНОВКИ

В епоху стрімкого технологічного розвитку та зростаючих потреб користувачів, технологія програмно-визначених мереж стає одним з провідних принципів для побудови сучасних мережевих інфраструктур. Застосування SDN у різних сферах свідчить про їхню виразну роль у вирішенні викликів, що постають перед сучасними мережами.

SDN розв'язує проблеми традиційних мереж, де об'єднання управління і передачі даних на одному пристрої призводить до обмежень у гнучкості та ефективності. Це дозволяє

централізованою програмній частині, контролеру, керувати роботою всіх мережевих пристроїв.

Проведені дослідження запропонованої реалізації для пріоритезації трафіку використання класифікатора на основі найкращого підходу до навчання моделі дозволить ефективно визначати пріоритети трафіку в режимі реального часу на тестовому стенді. Необхідно провести додаткові дослідження для оцінки масштабованості та впливу на мережевий трафік при різних умовах. Додаткові дослідження щодо масштабованості та впливу на мережевий трафік при різних умовах важливі для повного розуміння ефективності реалізації. Загалом, запропонований метод відкриває шлях до подальших досліджень та покращення, сприяючи вдосконаленню управління трафіком в програмно визначених мережах.

Слід враховувати, що ефективні стратегії пріоритезації трафіку вимагають комплексного підходу, зважаючи якість контролера, програмованість, тенденції у розвитку мереж та аспекти безпеки. Постійне вдосконалення та адаптація методів до зростаючих вимог та змін у технологічному середовищі є ключовими аспектами для забезпечення ефективності та стабільності системи в програмно визначених мережах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. O. Jefia, S. I. Popoola, i A. A. Atayero, «Software-defined networking: Current trends, challenges, and future directions», в *Proceedings of the International Conference on Industrial Engineering and Operations Management*, 2018.

2. R. Shubbar, M. Alhisnawi, A. Abdulhassan, i M. Ahamdi, «A Comprehensive Survey on Software-Defined Network Controllers», в *Lecture Notes in Networks and Systems*, 2021. doi: 10.1007/978-981-16-0666-3_18.

3. N. M. Kaliyamurthy, S. Taterh, S. Shanmugasundaram, A. Saxena, O. Cheikhrouhou, i H. Ben Elhadj, «Software-Defined Networking: An Evolving Network Architecture - Programmability and Security Perspective», *Security and Communication Networks*, вип. 2021, 2021, doi: 10.1155/2021/9971705.

4. S. S. Samaan i H. A. Jeiad, «Architecting a machine learning pipeline for online traffic classification in software defined networking using spark», *IAES International Journal of Artificial Intelligence*, вип. 12, вип. 2, 2023, doi: 10.11591/ijai.v12.i2.pp861-873.

ХМАРНИЙ СЕРВІС ДЛЯ ДИСТАНЦІЙНОГО ДОГЛЯДУ ЗА ПАЦІЄНТАМИ

Петренко А.І.¹, Редька М.Ю.²

Національний технічний університет України «Київський політехнічний інститут
ім. Ігоря Сікорського», Київ, Україна

¹ tolja.petrenko@mail.com, ² mex4232@gmail.com

Протягом останніх років спостерігається швидкий розвиток різноманітних технологій та методів для впровадження різноманітних веб-систем. Цей розвиток обумовлений, з одного боку, постійною інтеграцією нових систем у сферу веб, а з іншого боку – зростанням складності вже існуючих систем. Нові методи та технології призначені відповідати технологічним вимогам складних систем та полегшувати їх впровадження. Зокрема, галузь охорони здоров'я є однією з найбільш цікавих і водночас складних сфер. Зазвичай такі системи характеризуються високими технологічними та етичними стандартами, а також складними внутрішніми процесами, вимагаючи постійної підтримки та оновлень. Це призводить до природного інтересу до використання сучасних підходів для полегшення та покращення функціонування таких систем, які представлені в даній роботі.

Ключові слова: мікросервісна архітектура, мікрофронтенд, складні системи, охорона здоров'я.

1. ВСТУП

Хмарні обчислення традиційно використовувалися ІТ-індустрією для резервного копіювання даних, розробки та тестування програмного забезпечення, аварійного відновлення тощо. Однак із розвитком технологій постачальники хмарних послуг розробили різні рішення, адаптовані до конкретних галузей. Від охорони здоров'я та інтелектуального автомобілебудування до страхування, роздрібною торгівлі, маркетингу та освіти все більше галузей використовують хмару, щоб отримати переваги.

Хмарні обчислення в охороні здоров'я забезпечують зв'язане, доступне та спільне середовище для пацієнтів, докторів та клінік. У довгостроковій перспективі перехід на хмарне середовище забезпечить зниження витрат і сприятиме більшому комфорту для пацієнтів та медичного персоналу.

В охороні здоров'я хмарні обчислення набувають все більшої популярності, особливо після пандемії COVID-19. Важливість хмарних обчислень в медичній сфері очевидна з того факту, що дана галузь постійно розвивається та розширюється. Зараз хмарні обчислення є необхідною технологією для галузі охорони здоров'я, щоб забезпечити оптимальне обслуговування пацієнтів.

У цій статті пропонується архітектура додатку, що реалізує віртуальний кабінет лікаря та пацієнта, яка є ефективною з точки зору вирішення проблем комунікації лікарів та пацієнтів, а також відстеження процесу одужання хворих.

2. АРХІТЕКТУРА СИСТЕМИ ВІРТУАЛЬНОГО КАБІНЕТУ ЛІКАРТЯ ТА ПАЦІЄНТА

2.1. Вимоги до системи

Головна мета цієї системи полягає в полегшенні взаємодії між лікарем і пацієнтом під час розроблення та виконання плану лікування різних хвороб. Система повинна надавати лікарю зручний інтерфейс для створення плану лікування, його кроків і обмежень, а також забезпечувати пацієнту зручний спосіб перегляду плану та взаємодії з ним (включаючи можливість позначати виконані приписи, уточнювати план лікування та вводити дані про процедури в рамках плану).

Окрім цього базового функціоналу, система також повинна включати такі функції:

- Перегляд особистої інформації про пацієнта, зокрема його електронну медичну карту (ЕКМ).
- Моніторинг поточного стану пацієнта на основі введених даних.
- Автоматичні повідомлення про стан здоров'я на основі введених показників.
- Допоміжні інструменти для управління планом лікування, які допомагають лікарю в його управлінні та пацієнту в проходженні лікування.

Крім того, система повинна бути готовою для майбутнього розширення функціоналу, інтеграції з іншими сервісами та забезпечувати високий рівень конфіденційності даних користувачів.

В результаті роботи цієї системи пацієнти і лікарі отримують зручні інтерфейси, що містять всю необхідну інформацію та надають підказки для ефективного створення та виконання плану лікування.

З вищесказаного випливає, що система має багато різноманітних функцій, які можуть бути розподілені на різні компоненти, такі як:

- Менеджмент плану лікування.
- Зручні системи спілкування.
- Доступ до медичної історії та інформації про пацієнта.
- Аналітика стану пацієнта.
- Рекомендації для лікарів і пацієнтів.

В той же час в системі мають бути передбачені можливості для майбутнього розширення функціоналу системи, покращення окремих компонентів.

2.2. Опис архітектури

За встановленими умовами було розроблено архітектуру системи, яка включає в себе різні компоненти та забезпечує їх взаємодію, як то ілюструється на рис. 1.

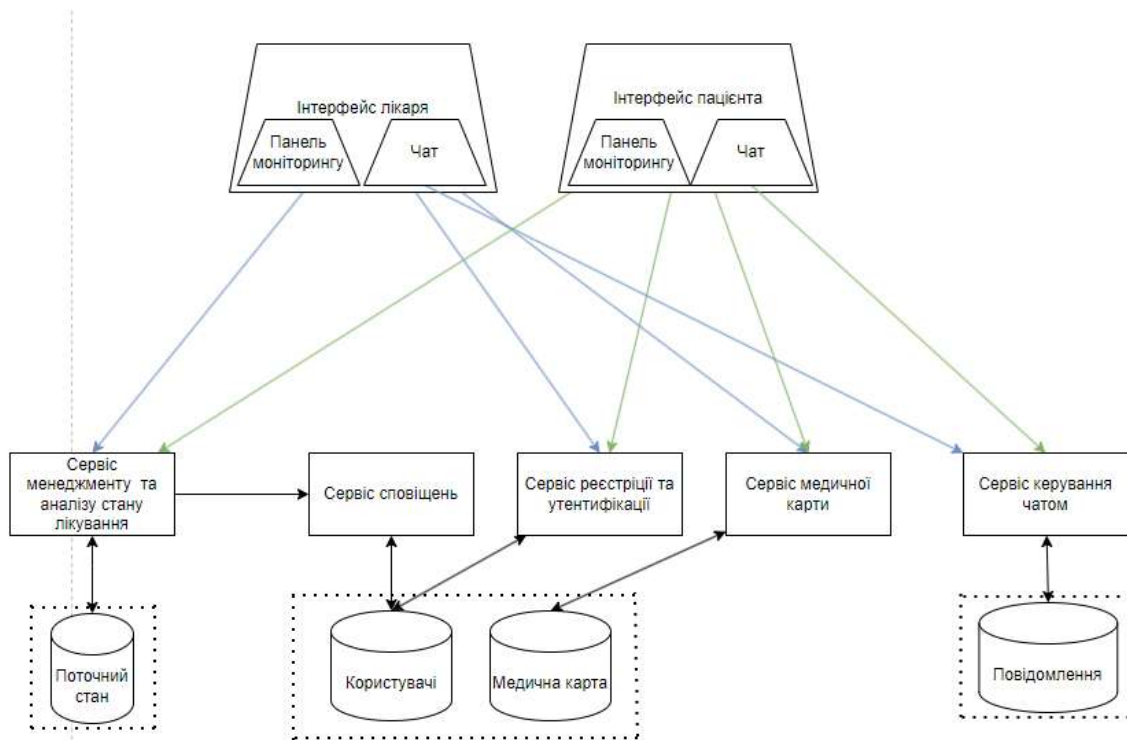


Рисунок 1. Архітектура системи віртуального кабінету лікаря та пацієнта

Значення компонентів системи:

Сервіс аналізу даних про поточний стан – безперервно перевіряє вміст відповідної бази даних і, у разі необхідності, видає сигнал для ініціації сповіщення або оновлення інформаційної панелі інтерфейсу.

Сервіс сповіщень – відповідає за розсилку сповіщень пацієнтам і лікарям щодо різних подій, включаючи кризові ситуації, за допомогою різних комунікаційних каналів, таких як СМС-повідомлення або електронна пошта. Додатковий інтерес представляють "допоміжні" або "аналітичні" сервіси.

Сервіс реєстрації, аутєнтифікації – є однією з ключових складових системи, відповідальною за належне управління політикою доступу до ресурсів, включаючи особисті дані.

Сервіс медичної картки – відповідає за надання особистої медичної інформації пацієнтів.

Сервіс керування чатом – забезпечує механізм спілкування лікаря\пацієнта.

Важливо відзначити деякі технічні особливості системи:

- "Чутлива" особиста інформація, яка може однозначно ідентифікувати користувача системи, зберігається в одному конкретному місці (на рис.1 відображений блок з елементами "Користувачі" і "Дані ЕМК"). Таким чином, ця інформація може бути окремо виведена в інший простір, навіть в окрему базу даних, яка обладнана додатковими заходами безпеки. Це може призвести до збільшення обсягу витрат ресурсів. Таблиці в інших просторах не містять прямих посилань на конкретних користувачів, тому в разі несанкціонованого доступу зловмисників до них, не вдається однозначно встановити, кому належать ці дані. Необхідні дані для зіставлення планів лікування, повідомлень та поточного стану доступні користувачам тільки в контексті "чутливих даних". Для наочності приклад відповідної організації таблиць приведено на рис. 2.

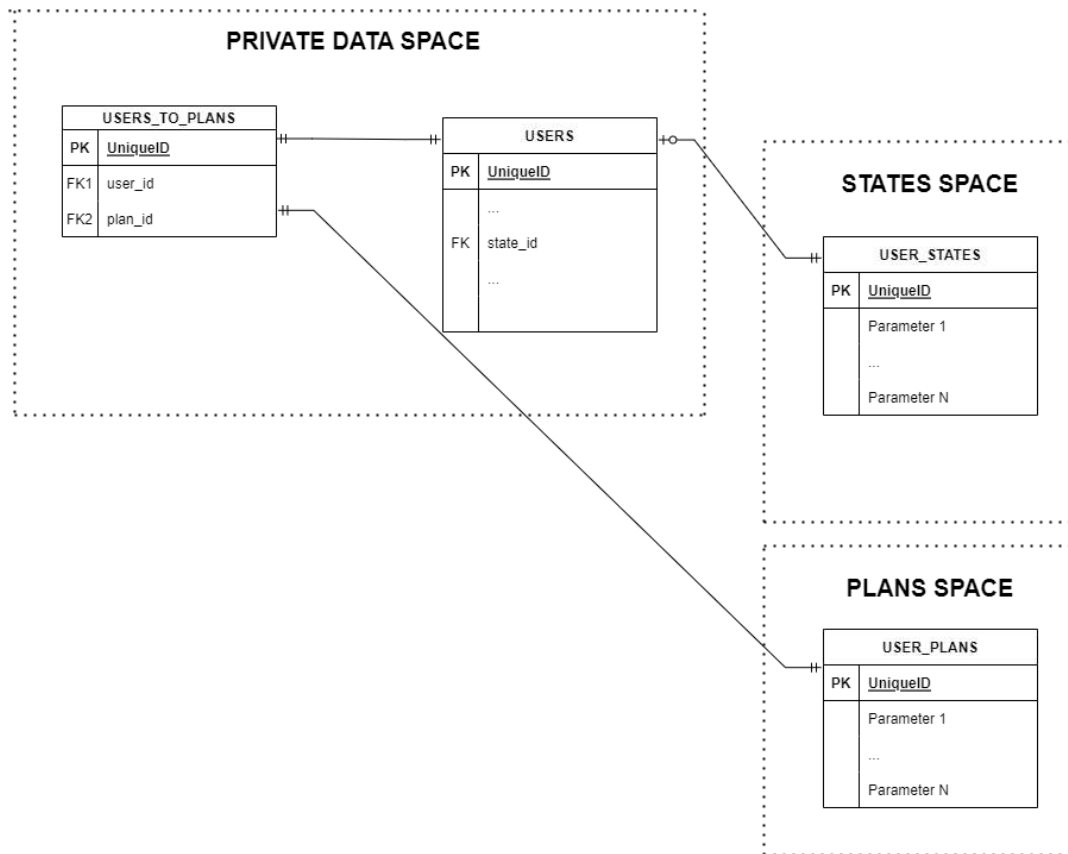


Рисунок 2. Відношення між таблицями з окремих просторів

- На рівні обслуговування або в базі даних можливо обмежити право деяким службам вносити зміни в певні таблиці. Наприклад, аналітичні сервіси обмежені можливістю лише зчитувати дані для подальшого аналізу, сервіс медичної картки може тільки надавати інформацію про картки, але не мати можливості змінювати їх, і лікар може не лише переглядати дані про поточний стан, але й змінювати їх.

- Оскільки таблиці, що описують поточні стани та плани лікування, фактично є "анонімними", існує можливість надати стороннім сервісам доступ для читання цих даних з метою виконання різноманітного практичного аналізу або для надання інформації зовнішнім статистичним сервісам.

2.3. Декомпозиція інтерфейсу

Підхід до розбиття на компоненти (мікросервіси) не обмежується внутрішньою структурою системи, а також застосовується на рівні користувацького інтерфейсу. Цей підхід називається "мікрофронтендним" [4, 5]. У даній системі він використовується для інтеграції з аналітичною панеллю, чатом і, в майбутньому, з іншими компонентами.

Цей підхід дозволяє спростити роботу команд, які працюють над одним проектом, і відкриває можливість використовувати послуги зовнішніх постачальників, а також використовувати підхід "white label" [6].

Зі схематичної точки зору сторінка плану лікування в такому випадку може виглядати, як показано на рис. 3. На ньому панель моніторингу (виділена синім кольором) та чат (виділений оранжевим кольором) імпортуються як окремі залежності ззовні і, за потреби, можуть бути вимкнені користувачами.

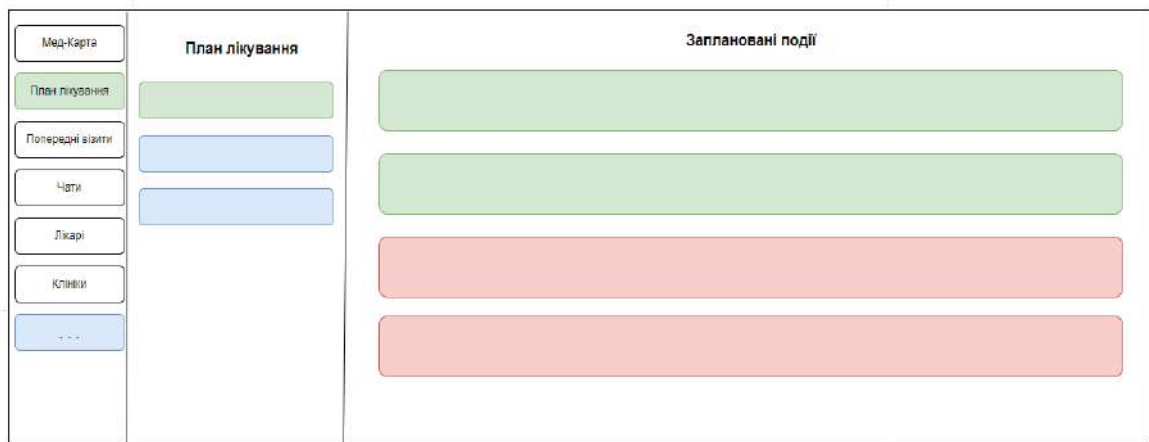


Рис. 3. Схематичний вигляд інтерфейсу, скомпонованого із різних частин (окремих субінтерфейсів)

3. ВИСНОВКИ

Хмарні обчислення впливають на галузь охорони здоров'я комплексно, з багатьох сторін. Важливість хмарних обчислень в охороні здоров'я величезна: від зниження вартості медичної допомоги до покращення зв'язку з пацієнтами та якості лікування. Вони роблять величезний позитивний вплив на компанії та клініки. Останні звіти показують, що впровадження хмарних рішень, безсумнівно, є посушним завданням для медичних організацій. Основними перешкодами є або проблеми безпеки, або брак внутрішніх технічних ресурсів.

У цьому дослідженні пропонується модель системної архітектури, яка зосереджена на розділенні та зниженні зв'язівок між компонентами не лише на серверній стороні (що вже є практикою), але й на стороні користувачього інтерфейсу (що ще не є стандартом). В запропонованій системі така декомпозиція дозволяє, з одного боку, ізолювати критичні дані та відповідні механізми роботи з ними, відокремлюючи їх від інших компонентів системи (для забезпечення надійності), і, з іншого боку, виділяти інші, менш важливі дані в окремі блоки, що відкриває можливість інтеграції з зовнішніми сервісами.

З огляду на особливості галузі, такий підхід до декомпозиції, включаючи сторону користувачього інтерфейсу, є досить обґрунтованим, оскільки системи в цій галузі часто функціонують "агрегатами" даних інших систем та джерел. Відповідно, модульний підхід дозволяє легко розширювати систему та налаштовувати зв'язки з зовнішніми системами, такими як постачальники медичної інформації, аналітичні системи та "white label" продукти, такі як чати та системи моніторингу.

Звісно, такий підхід до декомпозиції має свої витрати і труднощі в реалізації, але в даному випадку вони виправдані із-за значного обсягу функціоналу системи та вимог до безпеки, масштабованості та інтеграції. В той же час, він дозволяє підтримувати відносно не критичні вимоги до продуктивності та швидкодії системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Applications Of Cloud Computing In Healthcare URL: <https://www.ridge.co/blog/cloud-computing-in-healthcare/> (дата звернення: 23.10.2023).
2. Digital health (digital healthcare) URL: <https://www.techtarget.com/searchhealthit/definition/digital-health-digital-healthcare> (дата звернення: 12.10.2023).

3. REPORT on the use of cloud computing in health URL: https://health.ec.europa.eu/system/files/2018-02/ev_20151123_co06_02_en_0.pdf (дата звернення: 25.10.2023).
4. Digital health at the edge: Three use cases for the healthcare industry URL: <https://stlpartners.com/articles/edge-computing/digital-health-at-the-edge/> (дата звернення: 24.10.2023).
5. Cam, J. Micro Frontends. URL: <https://martinfowler.com/articles/micro-frontends.html> (дата звернення: 16.11.2023).
6. Amanda, L. White Label Software Explained: Benefits and Examples. URL: <https://growsurf.com/blog/white-label-software> (дата звернення: 18.11.2023).

ОРКЕСТРУВАННЯ ОБЧИСЛЕНЬ У МІКРОСЕРВІСНІЙ АРХІТЕКТУРІ

Полещук В.О.¹, Булах Б.В.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Кафедра системного проектування, Київ, Україна

¹ vladpoleshchuk@gmail.com, ² bogdan.bulakh@gmail.com

Це дослідження вивчає оркестрування обчислень в контексті мікросервісної архітектури, зосереджуючись на різних аспектах, таких як підходи до оркестрування в сервісно-орієнтованій архітектурі, інструменти для оркестрування, патерни оркестрування в мікросервісах, підходи до розробки оркестраторів мікросервісів, а також практична реалізація, що включає створення оркестратора, керованого подіями. Дослідження поєднує огляд літератури, аналіз інструментів та практичну реалізацію, щоб отримати уявлення про ефективні стратегії оркестрування. Результати дослідження відкривають нові перспективи в цій галузі, а практичний компонент пропонує зручний конструктор для оркестрування, керованого подіями, що підвищує зручність використання мікросервісів у реальних сценаріях.

Таким чином, це дослідження не тільки розкриває тонкощі оркестрування мікросервісів, але й робить відчутний внесок у вигляді зручного для користувача конструктора. Цей інструмент має практичне значення, пропонуючи спрощений шлях для розробників та організацій, які прагнуть використати можливості мікросервісів у послідовний та організований спосіб. Це дослідження відкриває можливості для подальшого вивчення і застосування в постійно мінливому ландшафті архітектури мікросервісів.

Ключові слова: оркестрація, мікросервіси, розробка оркестратора, інструментарій оркестрування, шаблони оркестрування.

1. ВСТУП

У постійно мінливому світі архітектури програмного забезпечення мікросервіси стали трансформаційною парадигмою, пропонуючи неперевершену гнучкість, масштабованість та оперативність у розробці додатків. Оскільки компанії все частіше переходять на мікросервіси, організація обчислень у цій розподіленій і модульній структурі стає критично важливою точкою координації. Цей звіт заглиблюється у багатогранну сферу оркестрування обчислень в архітектурі мікросервісів, досліджуючи різноманітні підходи, інструменти та патерни. Наше дослідження охоплює тонкощі оркестрування сервісів з метою розібратися в складнощах, вирішити проблеми та поділитися ідеями, які допоможуть архітекторам та розробникам орієнтуватися в динамічній екосистемі мікросервісів.

Архітектура мікросервісів, з її децентралізованим і модульним дизайном, вимагає ефективної оркестровки для безперебійної роботи. Оркестрування передбачає координацію взаємодії між розподіленими мікросервісами, і для цього процесу критично важливими є три ключові виміри: підходи, інструменти та патерни.

2. АКТУАЛЬНІСТЬ

Мікросервісна архітектура набула широкого розповсюдження завдяки своїй масштабованості, гнучкості та здатності сприяти швидкому розвитку. Однак, ефективна оркестрація обчислень в рамках цієї парадигми залишається важливою проблемою, що постійно розвивається. Оскільки організації все більше переходять на мікросервіси, потреба в ефективній оркестровці стає першочерговою для забезпечення безперебійної комунікації, масштабованості та відмовостійкості.

3. МЕТА ДОСЛІДЖЕННЯ

Це дослідження має на меті заглибитися в оркестровку обчислень в архітектурі мікросервісів, зосередившись на вирішенні поточних проблем та підвищенні ефективності взаємодії сервісів. Мета – надати практичні знання та методології, які допоможуть архітекторам і розробникам орієнтуватися в тонкощах оркестрування обчислень у динамічних середовищах мікросервісів, а також запропонувати інструмент для створення оркесторів на основі подій.

4. ПІДХОДИ ОРКЕСТРУВАННЯ

Сучасні підходи до оркестрування включають централізовані моделі, де центральний оркестратор координує мікросервіси, та децентралізовані стратегії, засновані на хореографії, де мікросервіси спільно взаємодіють без центрального контролера. Також з'явилися гібридні моделі, які поєднують централізовані та децентралізовані елементи. Ці підходи відрізняються компромісами, пов'язаними з контролем, масштабованістю та адаптивністю.

Business Process Execution Language, BPEL [1]:

BPEL, яку часто називають WS-BPEL (Web Services Business Process Execution Language), є стандартом для визначення та виконання бізнес-процесів на основі веб-сервісів. Він надає структуровану мову на основі XML для опису потоку сервісів та їх взаємодії. BPEL дозволяє моделювати складні бізнес-процеси, використовуючи такі конструкції, як дії, потоки та партнерські зв'язки.

Enterprise Service Bus (ESB) [2]:

Enterprise Service Bus – це технологія проміжного програмного забезпечення, яка виступає в якості центрального вузла для маршрутизації, трансформації та організації взаємодії сервісів. ESB надає набір готових роз'ємів та адаптерів, які дозволяють різним сервісам взаємодіяти один з одним.

Оркестрування на основі хореографії:

Приймає децентралізовану модель, де мікросервіси автономно взаємодіють без центрального контролера. Цей підхід спирається на спільні протоколи, що сприяє гнучкості, але вимагає чіткого розуміння міжсервісної комунікації.

Оркестрування, орієнтоване на людину:

Людино-орієнтована оркестрація, також відома як оркестрація робочого процесу, зосереджена на управлінні як автоматизованою взаємодією служб, так і людськими завданнями в рамках робочого процесу. Цей підхід підходить для процесів, які передбачають схвалення, рецензування або участь користувачів.

Оркестрування на основі правил:

Використовує заздалегідь визначені правила для управління потоком операцій. Оркестрування на основі правил забезпечує адаптивність, дозволяючи динамічно реагувати на мінливі умови на основі встановлених правил.

Динамічна оркестрація:

Підкреслює адаптивність і реагування на різні робочі навантаження. Динамічна оркестрація дозволяє в режимі реального часу вносити корективи у взаємодію сервісів і розподіл ресурсів, забезпечуючи оптимальну продуктивність системи.

5. ІНСТРУМЕНТИ ДЛЯ ОРКЕСТРУВАННЯ

Ефективна координація та управління розподіленими обчислювальними ресурсами вимагає складних інструментів, які можуть легко орієнтуватися в складнощах оркестрування. У цьому огляді досліджується та оцінюється спектр існуючих інструментів, призначених для оркестрування обчислень, від контейнерних платформ до безсерверних фреймворків та спеціалізованих рішень для оркестрування.

Kubernetes:

Платформа оркестрування контейнерів з відкритим вихідним кодом, що полегшує автоматизоване розгортання, масштабування та управління мікросервісами.

Docker Swarm:

Нативне рішення для кластеризації та оркестрування, що спрощує розгортання та управління мікросервісами.

Apache Mesos:

Універсальний менеджер кластерів, що абстрагує ресурси, придатний для оркестрування мікросервісів в масштабі в різних середовищах.

Nomad від HashiCorp:

Простий, але потужний оркестратор, розроблений для гнучкості та простоти використання при розгортанні та управлінні мікросервісами.

Service Grids:

Фреймворки, що забезпечують основу для сервіс-орієнтованих архітектур, полегшуючи оркестровку мікросервісів за допомогою виявлення сервісів та комунікації.

Платформи "Функція як послуга" (FaaS):

Забезпечують безсерверні обчислення, автоматизуючи виконання окремих функцій для полегшеного підходу до оркестрування мікросервісів.

Netflix Conductor:

Netflix Conductor – це механізм оркестрування з відкритим вихідним кодом, розроблений Netflix для полегшення координації мікросервісів у складних робочих процесах. Його основна мета – впорядкувати виконання бізнес-процесів, що робить його особливо ефективним для сценаріїв, коли послідовність мікросервісів повинна працювати разом для досягнення певної бізнес-цілі.

Власні інструменти оркестрування:

Індивідуальні рішення, розроблені власними силами для задоволення конкретних організаційних вимог і тонкощів оркестрування мікросервісів.

6. ШАБЛони ОКРЕСТРУВАННЯ

Основні патерни оркестрування в архітектурі мікросервісів включають в себе наступні [3]:

Централізована оркестрація:

Використовує центрального оркестратора для управління потоком діяльності та комунікації між мікросервісами.

Оркестрування на основі хореографії:

Використовує децентралізовану модель, де мікросервіси співпрацюють незалежно, покладаючись на спільні протоколи та неявні угоди.

Гібридна оркестровка:

Поєднує централізовані та децентралізовані підходи, адаптуючись до унікальних потреб різних сценаріїв мікросервісів.

Оркестрування на основі правил:

Використовує заздалегідь визначені правила для управління послідовністю та умовами взаємодії мікросервісів.

Динамічна оркестрація:

Надає пріоритет гнучкості та адаптивності, дозволяючи системі динамічно підлаштовуватися під мінливі умови та робочі навантаження.

На закінчення, оркестрування обчислень в мікросервісній архітектурі вимагає продуманого вибору підходів, інструментів і патернів. Динамічна і масштабована природа мікросервісів вимагає ретельного розгляду цих аспектів для забезпечення ефективних і відмовостійких розподілених систем.

7. ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ РОЗРОБКИ ОРКЕСТРАТОРІВ

Розробка оркестраторів включає такі аспекти, як вибір мови, архітектурний дизайн та стратегії інтеграції. Існуючі підходи охоплюють проксі-оркестри, сайдкари-оркестри, автономні оркестри, працюючі оркестри та безсерверні оркестри. Кожен підхід орієнтований на конкретні випадки використання та уподобання, надаючи розробникам різноманітні можливості для створення оркестраторів, пристосованих до їхньої архітектури мікросервісів. Серед найпопулярніших підходів:

Оркестратор на основі проксі

Оркестратор на основі проксі діє як посередник між мікросервісами. Він перехоплює зв'язок між сервісами, що дозволяє йому керувати такими аспектами, як балансування навантаження, маршрутизація та безпека.

Sidecar Orchestrator [4]

У підході на основі сайдкарів кожен мікросервіс поєднується з невеликим незалежним компонентом (сайдкаром), який виконує специфічні для оркестрування завдання. Це може включати виявлення сервісів, балансування навантаження і зв'язок з центральним оркестром.

Автономний оркестратор [4]

Автономний оркестратор – це спеціальний, незалежний сервіс, відповідальний за координацію мікросервісів. Він забезпечує централізовану точку контролю для управління взаємодією сервісів, виконанням робочих процесів та управлінням станом.

Оркестратор в процесі експлуатації

Оркестратори в процесі експлуатації вбудовують логіку оркестрування безпосередньо в кожну мікрослужбу, що дозволяє їм керувати власними взаємодіями та робочими процесами. Цей підхід іноді називають "мікро-оркестровкою".

Безсерверний оркестратор [5]

Безсерверні обчислювальні платформи також можуть слугувати оркестраторами для мікросервісів. Вони забезпечують керовану подіями оркестровку за допомогою функцій, які виконуються у відповідь на події або тригери.

8. РОЗРОБКА ОРКЕСТРАТОРА НА ОСНОВІ ПОДІЙ

Пропонуючи новий інструмент в арсеналі оркестрування мікросервісів, Constructor for Event-Driven Orchestrator дозволяє користувачам легко оголошувати та налаштовувати сервіси. Використовуючи YAML-файл, користувачі можуть вказувати сервіси, визначати протоколи зв'язку (наприклад, черги, REST API), формулювати події та визначати послідовність, в якій сервіси обробляють ці події. Цей інструмент не тільки впорядковує процес оркестрування, але й покращує контроль і кастомізацію для користувача в екосистемі мікросервісів.

Структура реалізації:

Конфігурація YAML:

Користувачі визначають сервіси, протоколи зв'язку, події та порядок обробки у структурованому файлі YAML. Цей декларативний підхід спрощує конфігурацію та покращує читабельність.

Декларація сервісу:

Кожна служба декларується з основними атрибутами, такими як ім'я, кінцева точка і тип. Це створює фундаментальне розуміння ландшафту мікросервісів.

Протоколи зв'язку:

Користувачі формулюють протоколи зв'язку між сервісами, вказуючи, чи відбувається взаємодія через черги, REST API або інші канали зв'язку. Ця гнучкість пристосована до різноманітних архітектур мікросервісів.

Специфікація подій:

Події визначаються з відповідними атрибутами, забезпечуючи чітке розмежування тригерів для оркестрування. Сюди входять типи подій, формати даних і будь-яка контекстна інформація, важлива для взаємодії сервісів.

Порядок виконання сервісів:

Інструмент дозволяє користувачам чітко вказати порядок, в якому сервіси обробляють події. Ця ієрархічна послідовність забезпечує структурований потік, що відповідає тонкощам запланованої оркестровки.

Обробка помилок та ведення журналів:

Включення надійних механізмів обробки помилок і функцій журналювання забезпечує прозорість і відмовостійкість. Користувачі можуть визначати стратегії обробки помилок і мати доступ до детальних журналів для полегшення пошуку та усунення несправностей і моніторингу.

Динамічне масштабування:

Щоб задовольнити вимоги динамічного масштабування, конструктор може бути спроектований так, щоб легко адаптуватися до змін у кількості сервісів або екземплярів. Це гарантує, що інструмент реагує на розвиток архітектури мікросервісів.

Інтеграція з існуючими інструментами:

Конструктор повинен легко інтегруватися з існуючими інструментальними засобами розробки та розгортання. Сумісність з популярними фреймворками та платформами сприяє його прийняттю в різноманітних екосистемах мікросервісів.

Дотримуючись цих принципів реалізації, Constructor for Event-Driven Orchestrator стає універсальним і орієнтованим на користувача інструментом, що дозволяє розробникам легко і точно оркеструвати мікросервіси.

9. ПРОБЛЕМИ

Архітектура мікросервісів, пропонуючи масштабованість і гнучкість, створює складні проблеми в оркеструванні обчислень:

Координація послуг: Координація виконання та підтримка зв'язку в децентралізованому середовищі є складним завданням.

Децентралізований контроль: Досягнення послідовного контролю та забезпечення дотримання всіма сервісами визначених робочих процесів викликає труднощі.

Динамічне масштабування: Організація обчислень в умовах динамічного масштабування та коливань розподілу ресурсів є нетривіальним завданням.

Обробка помилок: Виявлення, ізоляція та відновлення після помилок у різних сервісах створює проблеми з відмовостійкістю.

Узгодженість даних: Управління узгодженістю та станом даних між мікросервісами, особливо в розподілених базах даних, є складним завданням.

Безпека: Послідовне впровадження та управління політиками безпеки в різних мікросервісах вимагає ретельної уваги.

Інтеграція інструментарію: Інтеграція різноманітних інструментів та фреймворків у середовищі мікросервісів ускладнює безперебійну інтеграцію робочих процесів.

Складність моніторингу: Ефективний моніторинг та спостережливість в розподіленій архітектурі мікросервісів є складним завданням, що вимагає передових рішень.

Вирішення цих проблем є ключовим для реалізації повного потенціалу мікросервісів у сучасній розробці додатків.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Singhal N., Sakthivel U., Raj P. SELECTION MECHANISM OF MICRO-SERVICES ORCHESTRATION VS. CHOREOGRAPHY. researchgate.net. URL: https://www.researchgate.net/profile/Neha-Singhal-5/publication/330960463_Selection_Mechanism_of_Micro-Services_Orchestration_Vs_Choreography/links/5ca474e7a6fdcc12ee8f3a80.

2. Menge F. Enterprise Service Bus. programm.froscon.org. URL: https://programm.froscon.org/2007/attachments/15-falko_menge_-_enterprise_service_bus.pdf.

3. Taibi D., Lenarduzzi V., Pahl C. Architectural Patterns for Microservices: a Systematic Mapping Study. bia.unibz.it. URL: <https://bia.unibz.it/esploro/outputs/conferenceProceeding/Architectural-Patterns-for-Microservices-A-Systematic-Mapping-Study/991005773017601241#file-0> (дата звернення: 25.10.2023).

4. Burns B. Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. O'Reilly Media, 2018. 166 с.

5. Fowler S. Production-Ready Microservices. O'Reilly Media, 2016. 172 с.

АНАЛІЗ РИЗИКІВ В ЗАДАЧАХ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Северин М.С.¹, Мухін В.Є.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

¹ severyn.maxim@gmail.com, ² v_mukhin@i.ua [0000-0002-1206-9131]

Дослідження спрямоване на аналіз ризиків у сфері інформаційної безпеки, зокрема в завданнях, пов'язаних із захистом комп'ютерної системи. Робота включає в себе дослідження даних щодо існуючих різновидів кібератак, зокрема на основі NSL-KDD датасету, огляд існуючих моделей механізмів виявлення ризиків, а також розробка нової моделі. Науковою новизною є використання нової комбінації методів машинного навчання та їх параметрів для систем виявлення вторгнень (IDS) на основі мережі та хосту за допомогою аномалій, виявлених у мережевому трафіку комп'ютерної системи.

Ключові слова: інформаційна безпека, методи оцінки ризиків, різновиди кібератак, моделі виявлення і прогнозування, аналіз проблем безпеки комп'ютерних систем, штучний інтелект, машинне навчання.

1. ВСТУП

Тема кібербезпеки є дуже актуальною в сучасному цифровому світі. З кожним днем все більше організацій, включаючи урядові структури, банки, медичні установи та промислові підприємства стикаються з викликами і загрозами, пов'язаними з безпекою їх інформаційних систем.

У зв'язку зі зростанням кількості технологічних засобів, використовуваних у роботі організацій, збільшується і ймовірність виникнення кібератак, витоку конфіденційної інформації, втрати даних та порушення цілісності систем. Такі інциденти можуть мати серйозні наслідки, включаючи фінансові збитки, втрату довіри клієнтів і навіть негативний вплив на безпеку громадськості.

Аналіз ризиків в задачах інформаційної безпеки стає необхідністю для виявлення потенційних загроз та визначення належних заходів захисту. Це включає ідентифікацію вразливостей систем, аналіз можливих загроз і визначення ймовірності їх виникнення, а також оцінку потенційних наслідків [1].

Дослідження в області оцінки ризиків в задачах інформаційної безпеки має велике значення для розробки ефективних стратегій захисту інформаційних систем. Це допомагає організаціям прогнозувати, попереджати та зменшувати проблеми, пов'язані зі зловживанням інформацією, забезпечуючи стійкість і надійність їх інфраструктури.

Загальні напрями досліджень в даній галузі включають розробку нових методів оцінки ризиків, вдосконалення інструментальних засобів для аналізу та моделювання, вивчення тенденцій у сфері кібербезпеки та розробку стратегій захисту. Розуміння загроз і впровадження ефективних заходів безпеки інформаційних систем є важливими компонентами сучасного світу, який все більше стає цифровим та залежним від інформаційних технологій.

Ця стаття поглиблюється у сучасні підходи аналізу ризиків безпеки комп'ютерних систем. У ній досліджуються нові техніки, дані та можливості. Розробка моделі механізмів виявлення ризиків захисту є ключовим аспектом наших досліджень.

2. АНАЛІЗ ПОТЕНЦІЙНИХ РИЗИКІВ БЕЗПЕКИ КОМП'ЮТЕРНИХ СИСТЕМ

Безпека комп'ютерної системи (КС) – це стан захищеності КС і мереж від атак зловмисників, які можуть призвести до несанкціонованого розкриття інформації, крадіжки або пошкодження обладнання, програмного забезпечення або даних, а також до порушення або неправильного спрямування послуг, які вони надають.

Загроза – це потенційний ризик використання вразливості для порушення безпеки та ймовірного пошкодження/порушення інформації/сервісу, що зберігається/пропонується в комп'ютерних системах або через канали зв'язку. Загроза комп'ютерній системі виникає тоді, коли порушується конфіденційність (запобігання доступу до неї несанкціонованих осіб), цілісність (неможливість модифікації без дозволу) та доступність (доступність на вимогу уповноважених осіб) інформації в системі. Таким чином, до загроз комп'ютерним системам загалом можна віднести будь-яку навмисну, ненавмисну або спричинену природним шляхом дію, що призводить до втрати/маніпуляції з даними або фізичного знищення апаратного забезпечення [2].

Серед найпопулярніших кібератак можна виділити наприклад:

- Атаки для збору інформації – це практика отримання зловмисником цінних даних. Ця дія повністю пасивна, оскільки не є явною атакою.
- Розподілені DoS-атаки (DDoS) - жертву атакують одночасно з великої кількості окремих систем. DDoS-атаки зазвичай здійснюються за допомогою бот-мереж. Бот-майстер – це зловмисник, який опосередковано атакує машину-жертву, використовуючи армію ботів. DDoS-атаки відбуваються, коли велика кількість скомпрометованих систем діє синхронно і координується під контролем зловмисника, щоб повністю виснажити її ресурси і змусити відмовити в обслуговуванні справжнім користувачам.
- Атаки на основі IoT. Проблемою Інтернету речей є його слабка захищеність, оскільки ці пристрої часто ігноруються при встановленні патчів безпеки, які створюють лазівки для зловмисників, що дозволяють їм захопити ці пристрої та проникнути в мережу. Атака на основі IoT – це будь-яка кібератака, яка включає використання жертвою IoT для проникнення в мережу шкідливого програмного забезпечення.
- Програми-вимагачі – це шкідливе програмне забезпечення, яке перешкоджає доступу до комп'ютера або файлів на ньому. Комп'ютери можуть бути заблоковані, а файли зашифровані. Від жертви вимагають викуп за зняття обмежень, і це відображається в системі жертви.
- Шпигунське та рекламне програмне забезпечення – це програми, які мають спільну властивість збирати особисту інформацію користувачів без їхнього відома. Шпигунське ПЗ включає клавіатурні шпигуни, які реєструють усе, що набирається на клавіатурі, і створюють високий ризик викрадення особистих даних.
- Спам - це небажані масові повідомлення електронної пошти, які дратують користувача надмірною кількістю небажаних листів. Це створює навантаження для постачальників послуг зв'язку, організацій та приватних осіб. Спам також використовується як інструмент для фішингу.
- Атаки на веб-сайти – націлені на компоненти браузера, які можуть бути незахищеними, навіть якщо вони встановлені в браузері. Атаки з використанням SQL-ін'єкцій спрямовані на будь-який веб-сайт або веб-додаток, що використовує базу даних SQL, таку як MySQL, Oracle тощо, шляхом використання недоліків безпеки в програмному забезпеченні додатку [3–5].

Класифікація ризиків інформаційної безпеки дозволила систематизувати можливі загрози та їх характер, що є важливим етапом для подальшого аналізу та захисту інформації. Аналіз ризиків інформаційної безпеки виявився складним процесом, який передбачає врахування багатьох факторів, таких як ймовірність впливу та характер загроз, який не завжди легко оцінити чисельно.

3. АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ ВИЯВЛЕННЯ РИЗИКІВ БЕЗПЕКИ В КС

Для виявлення загроз/вторгнень у комп'ютерну систему існує багато мережевих сканерів або сканерів портів (Nessus, OpenVAS, Qualys, Rapid7Nexpose, Metasploit). Аналіз системи на основі мережевих та хост-даних є важливою складовою безпеки комп'ютерних систем. Загальний принцип заключається в тому, що мережевий трафік відстежується та витягує зашифрований зв'язок у формі файлів захоплення пакетів. При цьому використовується трафік HTTP, оскільки це найпоширеніший протокол для глобального зв'язку. Заголовки HTTP містять дані, які можна використовувати для виявлення зловмисної поведінки. Щоб дістати ці дані, можна проаналізувати потоки TCP із трасуванням HTTP із файлів захоплення пакетів (PCAP) або іншим аналізатором (Wireshark, Zeek).

Система, яка називається системою виявлення вторгнень (Intrusion detection system, IDS), відстежує мережевий трафік на наявність зловмисних транзакцій і негайно надсилає сповіщення, коли це спостерігається. Це програмне забезпечення, яке перевіряє систему на наявність зловмисних дій або порушень політики. Кожна незаконна діяльність або порушення часто фіксується централізовано за допомогою системи SIEM або повідомляється адміністрації.

IDS бувають 5 типів:

- Система виявлення вторгнень у мережу (NIDS): Системи виявлення вторгнень у мережу (NIDS) встановлюються в запланованій точці мережі для перевірки трафіку з усіх пристроїв у мережі. Він виконує спостереження за трафіком, що проходить у всій підмережі, і зіставляє трафік, який передається підмережами, із колекцією відомих атак. Після виявлення атаки або виявлення ненормальної поведінки сповіщення можна надіслати адміністратору. Прикладом NIDS є його встановлення в підмережі, де розташовані брандмауери, щоб побачити, чи хтось намагається зламати брандмауер.
- Система виявлення вторгнень (HIDS): Системи виявлення вторгнень (HIDS) працюють на незалежних хостах або пристроях у мережі. HIDS відстежує лише вхідні та вихідні пакети з пристрою та сповіщатиме адміністратора, якщо буде виявлено підозрілу або зловмисну активність. Він робить знімок існуючих системних файлів і порівнює його з попереднім знімком. Якщо файли аналітичної системи були відредаговані або видалені, сповіщення надсилається адміністратору для дослідження. Приклад використання HIDS можна побачити на критично важливих машинах, на яких не очікується зміна компонентів.
- Система виявлення вторгнень на основі протоколу (PIDS): Система виявлення вторгнення на основі протоколу (PIDS) складається з системи або агента, який постійно розміщується на передній частині сервера, контролюючи та інтерпретуючи протокол між користувачем/пристроєм і сервером. Він намагається захистити веб-сервер, регулярно відстежуючи потік протоколу HTTPS і приймаючи відповідний протокол HTTP. Оскільки HTTPS не зашифрований і перед миттєвим переходом на рівень веб-презентації, ця система повинна знаходитися в цьому інтерфейсі між використанням HTTPS.

- Система виявлення вторгнень на основі протоколу додатків (APIDS): Система виявлення вторгнень на основі протоколу додатків (APIDS) – це система або агент, який зазвичай знаходиться в групі серверів. Він визначає вторгнення, відстежуючи та інтерпретує зв'язок за протоколами, що стосуються окремих програм. Наприклад, це явно відстежуватиме протокол SQL для проміжного програмного забезпечення під час його взаємодії з базою даних на веб-сервері.
- Гібридна система виявлення вторгнень: гібридна система виявлення вторгнень створюється шляхом поєднання двох або більше підходів до системи виявлення вторгнень. У гібридній системі виявлення вторгнень дані агента хоста або системи поєднуються з мережевою інформацією для створення повного уявлення про мережеву систему. Гібридна система виявлення вторгнень більш ефективна в порівнянні з іншими системами виявлення вторгнень. Prelude є прикладом Hybrid IDS [6, 7].

4. РОЗРОБКА МОДЕЛІ

Згідно усього вище сказаного, було запропоновано архітектуру системи, що містить набір компонентів та забезпечує взаємодію між ними, як показано на рисунку 1.

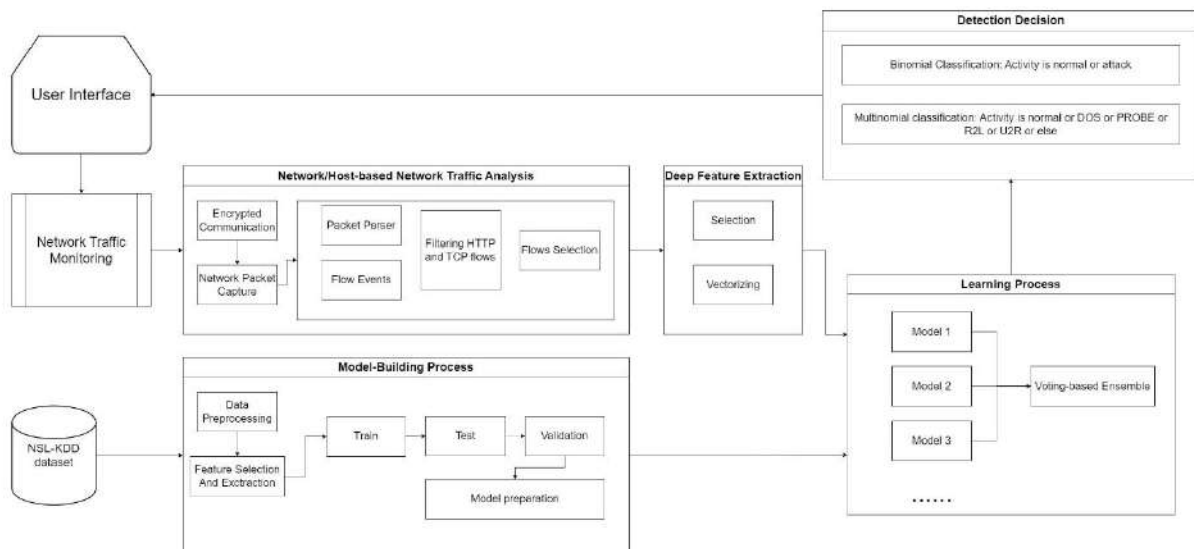


Рисунок 1. Архітектура програми

В інтерфейсі користувача можна запускати сканування мережі, а також отримувати результати після завершення. Блок Network Traffic Monitoring відповідає за збір даних.

- Шифрована комунікація (Encrypted Communication) – трафік мережі шифрується, щоб захистити вміст від несанкціонованого доступу. Існують методи аналізу метаданих і інших аспектів для отримання корисної інформації.
- Захоплення пакетів мережі (Network Packet Capture) включає в себе запис пакетів мережі для подальшого аналізу. Це дозволяє детально вивчати взаємодію на мережі, включаючи дані, що передаються між пристроями.
- Аналізатор пакетів (Packet Parser) використовується для розбору (аналізу) записаних пакетів мережі. Парсери можуть виділяти різноманітні дані, такі як заголовки пакетів, адреси, протоколи тощо.
- Події потоків (Flow Events) - вказує на виникнення подій або змін в потоках мережевого трафіку.

- Фільтрація HTTP і TCP потоків (Filtering HTTP and TCP Flows) – означає вибірку обробку потоків, пов'язаних з протоколами HTTP і TCP. Це дозволяє виділяти конкретний тип трафіку для подальшого аналізу.
- Вибір потоків (Flow Selection) полягає в обранні конкретних мережевих потоків для подальшого аналізу. Це може бути корисним для зосередження на конкретних подіях або пристроях.

Deep Feature Extraction – відповідає за вибір та (можливо) за векторизацію даних, отриманих скануванням. Не всі дані можуть бути потрібними для аналізу, а також для деяких нейронних мереж їх початковий вигляд може бути неприйнятним без векторизації [8].

Model-building process – побудова моделей машинного навчання відбувається з використанням NSL-KDD датасета. NSL-KDD (Network-based Intrusion Detection Dataset – KDD Cup 99) є датасетом, який використовується для розробки та тестування систем виявлення вторгнень у мережі (IDS). Він є розширенням відомого датасету KDD Cup 99, який був використаний для конкурсу KDD Cup 1999 щодо виявлення вторгнень.

NSL-KDD містить п'ять основних класів трафіку:

- Normal – легітимний мережевий трафік, який вважається безпечним.
- DoS (Denial of Service) – атаки, спрямовані на перевантаження ресурсів, забираючи їх від легітимних користувачів.
- Probe – атаки, які включають сканування мережі з метою виявлення слабких місць для подальших атак.
- U2R (User to Root) – атаки, спрямовані на отримання прав адміністратора системи з прав користувача.
- R2L (Remote to Local) – атаки, які спрямовані на отримання доступу до системи через мережу, якщо ви вже є користувачем на цій системі.

NSL-KDD містить різноманітні атаки та легітимний трафік для створення більш реалістичного середовища для виявлення вторгнень. Він став популярним інструментом для тестування та порівняння алгоритмів виявлення вторгнень в мережах.

Learning process – компонент, що визначає механізм вивчення обраними ML моделями даних, отриманих на основі згенерованого трафіка. Ensemble – це стійка модель, створена шляхом системного поєднання базових технологій. На відміну від індивідуальних моделей, модель ансамблю здатна вирішувати проблеми класифікації та регресії. У запропонованому дослідженні використовується ансамблевий підхід м'якого опитування. Для початку ми використали навчальні дані для створення основних моделей. А потім їх ефективність перевіряється за допомогою тестових даних, причому кожна модель створює унікальну класифікацію. Щоб отримати остаточну класифікаційну продуктивність, ансамблеве навчання використовує оцінки кількох підходів як додаткову інформацію.

Detection Decision – може працювати у двох режимах: або просто помічати аномальну активність (виявлення), або визначати ще й характер атаки (прогнозування). Для цього застосовується біноміальна та поліноміальна класифікації. Результат користувач отримує в інтерфейсі програми [9].

5. ВИСНОВКИ

Запропонована архітектура системи для аналізу ризиків порушення захищеності в КС містить комплекс компонентів, що взаємодіють для повноцінної роботи. Основні компоненти включають захоплення пакетів мережі, аналізатор пакетів, виявлення подій потоків, фільтрацію HTTP і TCP потоків, вибір потоків, витягування глибоких ознак, процес побудови моделей машинного навчання з використанням NSL-KDD датасету та ансамбль-навчання.

NSL-KDD датасет використовується для розробки та тестування систем виявлення вторгнень у систему, містить різноманітні класи трафіку, включаючи атаки DoS, Probe, U2R, R2L, а також легітимний трафік. Застосування ансамбль-навчання дозволяє покращити класифікаційну продуктивність, використовуючи оцінки різних моделей як додаткову інформацію. Результати аналізу можуть використовуватися для виявлення аномалій або навіть для передбачення характеру атаки.

Однак важливо враховувати, що жодна система не є абсолютною та повністю невразливою. Запропонована архітектура забезпечує важливі етапи аналізу та захисту, але враховувати інші аспекти, такі як соціальна інженерія та внутрішні загрози, також є критичним. Загальна безпека вимагає комплексного підходу, який об'єднує технічні та людські аспекти.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Information Security Risk Analysis, Second Edition, Thomas R. Peltier URL: https://www.academia.edu/40140524/Information_security_risk_analysis_thomas_r_peltier
2. Ievgeniia Kuzminykh , Bogdan Ghita, Volodymyr Sokolov Security Risk Assessment. URL: https://www.researchgate.net/publication/353436973_Information_Security_Risk_Assessment#fullTextFileContent
3. Mouna Jouinia Classification of security threats in information systems. URL: https://www.researchgate.net/publication/315714820_Classification_of_security_threats_in_information_systems#fullTextFileContent
4. Analysis of Computer Network Security Problems and Countermeasures. URL: https://www.researchgate.net/publication/320177233_Analysis_of_Computer_Network_Security_Problems_and_Countermeasures
5. Comparative analysis of modern methods of cryptographic protection of information URL: https://www.academia.edu/35708980/Comparative_analysis_of_modern_methods_and_algorithms_of_cryptographic_protection_of_information
6. Intrusion Detection System (IDS). URL: <https://www.geeksforgeeks.org/intrusion-detection-system-ids/>
7. Different types of Intrusion Detection Systems (IDS). URL: <https://wisdomplexus.com/blogs/different-types-of-intrusion-detection-systems-ids/>
8. Bisyrion Wahyudi, Kalamullah Ramli, and Hendri Murfi Implementation and Analysis of Combined Machine Learning Method for Intrusion Detection System. URL: https://www.researchgate.net/publication/326983606_Implementation_and_Analysis_of_Combined_Machine_Learning_Method_for_Intrusion_Detection_System
9. Cyber-Risk Forecasting using Machine Learning Models and Generalized Extreme Value Distributions. URL: <https://hal.science/hal-03814979/document>

ПЛАТФОРМИ З НИЗЬКОЮ ЗАТРИМКОЮ ДЛЯ ОБРОБКИ ПОТОКІВ ДАНИХ У РЕАЛЬНОМУ ЧАСІ

Сіркович А.І.¹, Харченко К.В.

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Інститут прикладного системного аналізу, Київ, Україна

¹ sirkovych.arsen@iit.kpi.ua

Стаття розглядає "платформи обробки поточкових даних з низькою затримкою" як складову розроблення. У тексті аналізуються стратегії та методи використання таких платформ, проводиться порівняльний аналіз популярних зразків цього типу. Дослідження виявляє нові можливості та переваги використання платформи для обробки поточкових даних у розробці, підкреслюючи її науковий та практичний внесок у галузь програмного забезпечення.

Ключові слова: наукова конференція, доповідь, сервіс, низька затримка, поточкові дані.

1. ВСТУП

Впровадження інформаційних систем, здатних обробляти великі обсяги даних в реальному часі, стало актуальною проблемою в епоху цифрової трансформації. За зростання обсягів даних з різних джерел виникає потреба в створенні ефективних і високопродуктивних інформаційних систем. Проте викликом є необхідність обробки цих даних з низькою затримкою для оперативного реагування та прийняття обґрунтованих рішень в реальному часі.

Робота буде фокусуватися на дослідженні аспектів цієї теми, включаючи аналіз сучасних високопродуктивних платформ для обробки поточкових даних, методи зниження затримки та підвищення продуктивності, а також розробку практичних застосувань для інформаційних систем в реальному часі.

2. ЩО TAKE LOW-LATENCY STREAM PROCESSING PLATFORMS?

Платформи з низькою затримкою для обробки потоків даних у реальному часі (Low-Latency Stream Processing Platforms) – це програмні рішення або середовища, які призначені для обробки поточкових даних з низькою затримкою, тобто з мінімальним часом між моментом створення даних і їх обробкою. Ці платформи використовуються в сценаріях, де важлива швидка реакція на надходження даних, такі як аналіз реального часу, моніторинг, прогнозування та інші застосування, де обробка даних повинна відбуватися негайно.

Їхні переваги полягають у здатності забезпечувати реальний час в обробці даних, забезпечуючи миттєву реакцію на зміни у навколишньому середовищі. Це важливо для застосунків, де необхідно оперативно реагувати на події та приймати швидкі та обґрунтовані рішення.

Додатково, платформи цього типу відзначаються високою швидкістю обробки великих обсягів даних, що дозволяє використовувати останні дані для аналізу та виведення результатів. Вони підтримують горизонтальну масштабованість, що дозволяє легко розширювати систему для роботи з ростом обсягу інформації.

Ці платформи також володіють модульністю та гнучкістю, що спрощує їх інтеграцію з іншими технологіями та системами. Можливість виконувати поточкову аналітику на льоту дозволяє отримувати актуальну інформацію для прийняття рішень.

Взагалі, платформи з низькою затримкою для обробки потоків даних у реальному часі дозволяють розробникам створювати високоефективні та гнучкі системи, що забезпечують негайну реакцію на зміни в навколишньому середовищі та динамічно взаємодіють з потоками інформації.

3. ПЕРЕВАГИ ВИКОРИСТАННЯ ТЕХНОЛОГІЇ

Використання платформ з низькою затримкою для обробки потоків даних у реальному часі дарує значні переваги в розробці різноманітних систем та застосунків. Ці платформи, спрямовані на миттєву обробку даних без затримок, стають ключовим інструментом у сучасних областях, де важлива оперативність та надійність систем.

Однією з головних переваг є можливість забезпечити реальний час у відповіді на події. Це означає, що система може миттєво реагувати та приймати рішення, що важливо для застосунків, де необхідна швидка реакція на зміни у середовищі.

Також слід відзначити оптимізацію продуктивності системи. Платформи з низькою затримкою дозволяють ефективно обробляти великі обсяги даних в реальному часі, надаючи можливість використовувати найновішу інформацію для прийняття рішень та виконання завдань.

Масштабованість є ще однією перевагою, оскільки систему можна легко розширювати за допомогою додаткових ресурсів для впорядкування з ростом обсягу даних та завдань обробки. Модульність і гнучкість також важливі, дозволяючи легко інтегрувати платформу з іншими інструментами та системами, що сприяє швидкій розробці та розгортанню різноманітних застосунків.

Крім того, можливість виконувати потокову аналітику, аналізуючи дані на льоту, полегшує отримання актуальної інформації. Ефективне використання ресурсів, оптимізація обчислювальних ресурсів та автоматизація оптимізації процесів також сприяють успішній розробці систем.

Загальною вагомою перевагою є те, що платформи з низькою затримкою для обробки потоків даних дозволяють розробникам створювати реактивні, ефективні та гнучкі системи, які відповідають вимогам сучасних динамічних інформаційних середовищ.

4. АРХІТЕКТУРА ПЛАТФОРМИ

Розглянемо структуру платформи для обробки потоків даних у реальному часі на прикладі Apache Kafka (рис. 1). Архітектура включає в себе кілька ключових елементів. Розглянемо їх детальніше.

Виробник (Producer). Цей компонент відповідає за генерацію та відправку потоку даних до Kafka-кластера. Виробник може виробляти дані з різних джерел та надсилати їх до одного або кількох тем Kafka.

Теми (Topics). Теми є логічними категоріями, які дозволяють організувати дані в Kafka. Кожен потік даних призначається певній темі, щоб інші компоненти могли підписатися на цю тему та отримувати відповідні дані.

Брокери (Brokers). Брокери відповідають за прийом, зберігання та розподіл поточкових даних. Кластер Kafka складається з декількох брокерів, які співпрацюють для забезпечення масштабованості та надійності.

Споживачі (Consumer). Споживачі підписуються на теми та обробляють потік даних, які надходять від виробників. Споживачі можуть бути організовані в групи для ефективної розподіленої обробки даних.

Групи споживачів (Consumer Groups). Групи споживачів дозволяють масштабувати споживання даних, розділяючи завдання між споживачами. Кожна група обробляє частину повідомлень, що дозволяє досягти паралельності та високої продуктивності.

Зберігання (Storage). Деякі дані можуть зберігатися в Kafka для подальшого використання або відновлення під час відмов. Дозволяє зберігати дані на задовільний термін для аналізу або відновлення стану системи.

Зокіп (ZooKeeper). ZooKeeper використовується для управління та координації компонентів Kafka. Забезпечує служби для виявлення та відновлення збоїв, а також для керування конфігурацією кластера.

Ця структура дозволяє Apache Kafka забезпечувати ефективну обробку потоків даних у реальному часі, масштабування та високу надійність в середовищі з низькою затримкою.

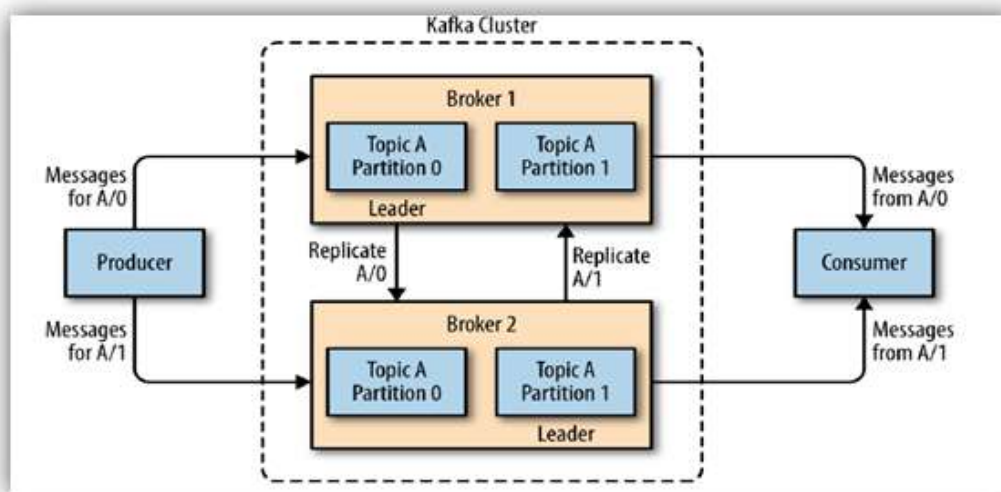


Рисунок 1. Приклад архітектури

5. ВИСНОВКИ

Дана робота, присвячена платформам з низькою затримкою для обробки потоків даних у реальному часі, розглянуто важливі аспекти цього передового напрямку в інформаційних технологіях. Аналізуючи такі платформи, як Apache Kafka, були визначені та проаналізовані ключові компоненти їхньої архітектури, включаючи виробників, споживачів, та механізми обробки даних.

Дослідження вказує, що платформи з низькою затримкою грають критичну роль у розвитку інформаційних систем, здатних працювати з великими потоками даних в режимі реального часу. Вони забезпечують високу швидкість обробки, гнучкість у взаємодії з різними джерелами даних, надійність та можливість аналізу інформації в реальному часі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “ O’Sullivan, M., & Perry, M. (2019). Real-time Big Data Analytics: Emerging Architecture. *Journal of Computer Science and Technology*, 20(3), 437-452.
2. Smith, J., & Johnson, L. (2020). Stream Processing Systems: A Comparative Study. *International Journal of Data Science and Analytics*, 15(2), 187-204.
3. Brown, A., & Davis, R. (2018). Low-Latency Data Processing with Apache Kafka. *Journal of Distributed Computing*, 25(1), 89-102.
4. Gorman, T. (2017). *Real-time Analytics with Storm and Kafka*. O'Reilly Media.

СЕРВІС ВЕДЕННЯ ФІНАНСОВОЇ ЗВІТНОСТІ ДЛЯ ФОП

Скрипник А.В.¹, Булах Б.В.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Кафедра системного проектування, Київ, Україна

¹ programworkerbox@gmail.com, ² bogdan.bulakh@gmail.com

Основною метою дослідження є пошук шляхів побудови мінімалістичної, але гнучкої у налаштуванні системи автоматизації звітності для ФОП з урахуванням можливості розширення до інших форм підприємств. У реалізації, що пропонується, реалізовано функції для ведення ФОП 3 групи з можливістю CRUD операцій для фінансових записів та складання поквартальних податкових звітностей з обчисленням єдиного соціального внеску.

Ключові слова: SaaS, сервіс фінансової звітності, розподілена система, графічний інтерфейс

1. ВСТУП

В сучасному підприємницькому середовищі, де правила та вимоги до фінансової звітності постійно змінюються, роль ефективного бухгалтерського сервісу для фізичних осіб-підприємців (ФОП) в Україні стає надзвичайно важливою. Забезпечення правильного ведення бухгалтерського обліку та своєчасної подачі податкових звітів не лише дозволяє уникнути можливих штрафів та санкцій, але й є ключовим фактором для стабільного фінансового управління та успішної діяльності підприємців.

У зв'язку з постійними змінами в законодавстві та високим рівнем відповідальності перед податковими органами, ФОП повинні мати доступ до кваліфікованого бухгалтерського супроводу. Це стає ключовим елементом ефективного управління фінансами та може визначити конкурентоспроможність та стійкість бізнесу в умовах сучасного ринкового середовища.

2. ОПИС СИСТЕМИ BOOKKEEPING AS A SERVICE

Пропонується реалізувати систему звітності як мережевий застосунок, що складається з таких ланок: графічний web-інтерфейс, модуль бухгалтерських обчислень та реляційна база даних. Діаграму викликів модулів під час роботи системи подано на рис. 1.

2.1. Модуль графічного інтерфейсу

Модуль графічного інтерфейсу реалізований мовою Python за допомогою бібліотеки streamlit [1]. Він реалізує взаємодію користувача із системою та використовує автентифікацію Google [2] для надання доступу різним користувачам. Тут зберігаються токени для користування Google API для певного користувача та результати деяких запитів до модуля бухгалтерських обчислень для зменшення навантаження на нього. Приклад вигляду сторінки складання звітів подано на рис. 2. В результаті використання зовнішнього сервісу для автентифікації користувачів, уникнено необхідності зберігання паролів та їх верифікації, що підсилює безпеку системи до зламів паролів.

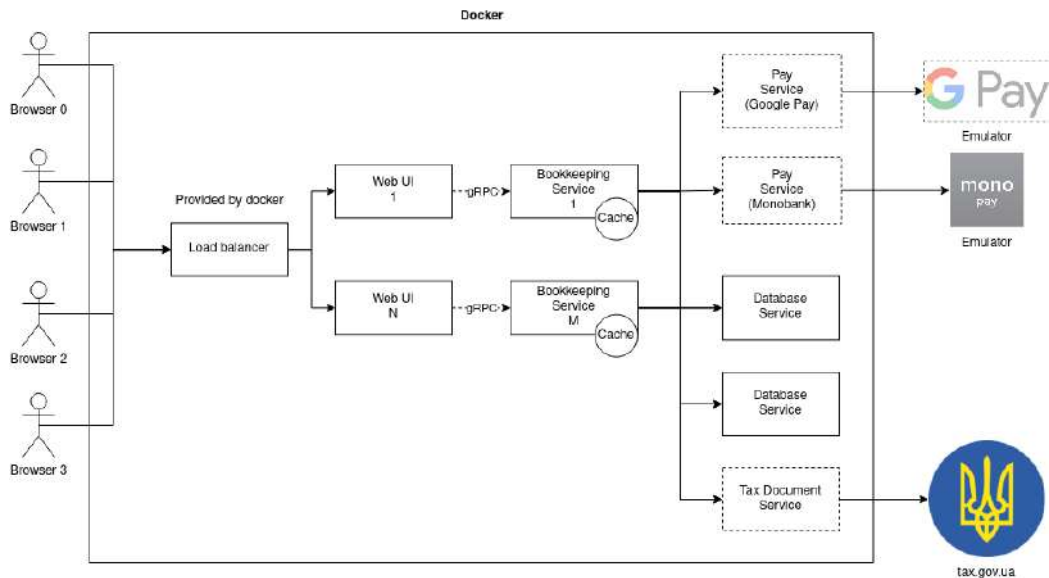


Рисунок 1. Діаграма викликів модулів системи Bookkeeping as a Service

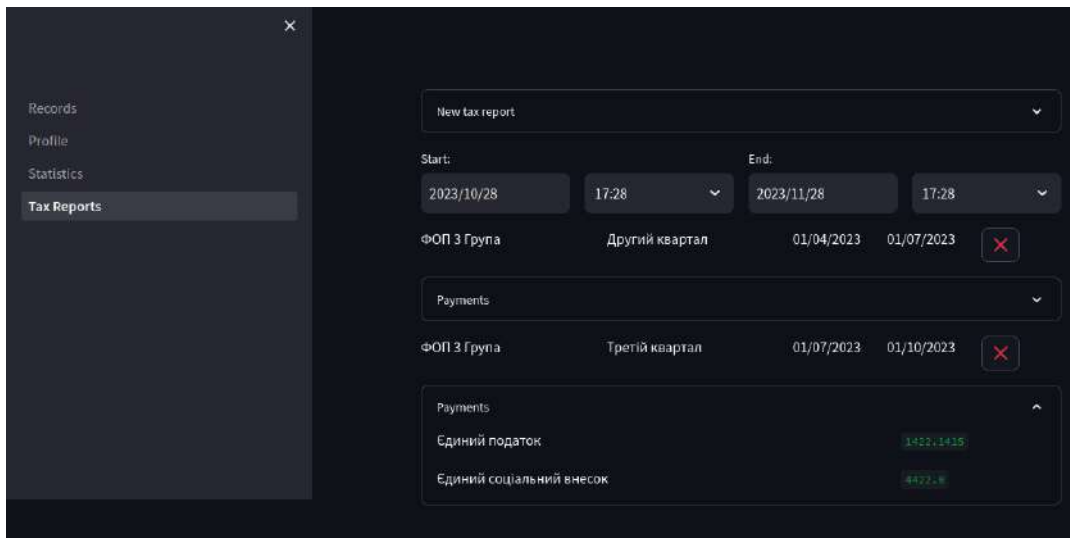


Рисунок 2. Вигляд сторінки подання звітів в системі Bookkeeping as a Service

2.2. Модуль бухгалтерських обчислень і взаємодії з базою даних

Основні операції з даними користувачів виконує модуль бухгалтерських обчислень. Він реалізує протокол GRPC [3] для взаємодії з модулем графічного інтерфейсу. Для цього використовується окремий пакет Python з визначенням інтерфейсів для взаємодії між модулями, який використовується як графічним інтерфейсом для формування запитів так і модулем бухгалтерських обчислень. Він генерує відповідні файли Python за допомогою інтерфейсу сервісу, описаному за допомогою Protobuf [4].

Також цим модулем виконуються запити до бази даних PostgreSQL [5] за допомогою object-relational mapping (ORM) бібліотеки SQLAlchemy [6]. Діаграма послідовностей для обробки запиту користувача подана на рис. 3.

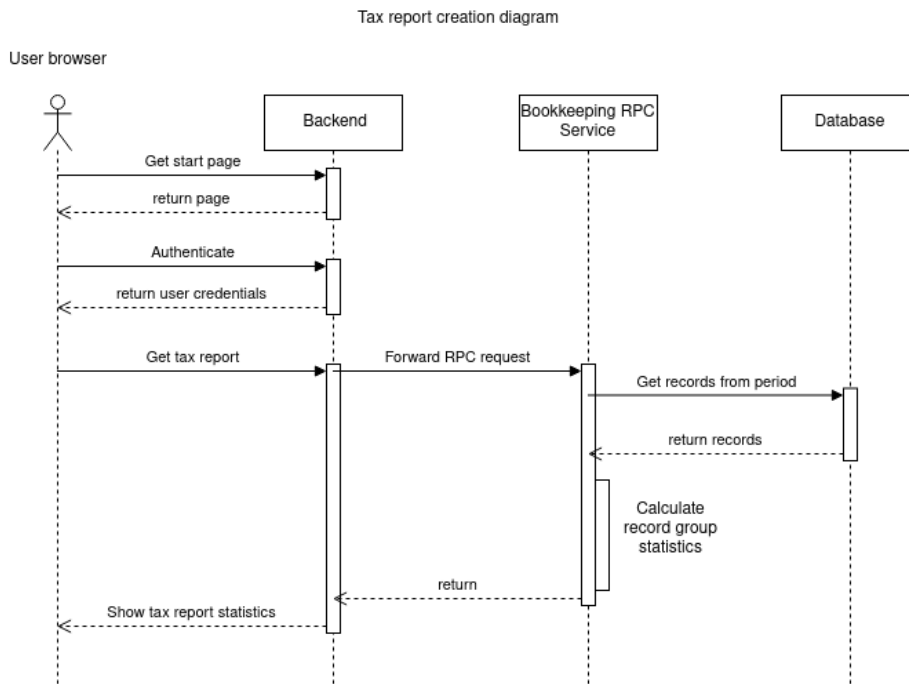


Рисунок 3. Діаграма послідовностей для обробки запиту на отримання користувачем статистики за записами в системі Bookkeeping as a Service

Основний функціонал додатку, пов'язаний зі створенням податкових звітностей з певною формою підприємства за конкретні періоди реалізується за допомогою загального інтерфейсу для всіх класів податкових звітностей. Для більш гнучкого оновлення логіки обробки фінансових записів відповідно до законодавства, реалізації фінансових звітностей подаються окремим модулем python, що може бути оновлений до новішої версії без зміни основної логіки додатку, за рахунок незмінного спільного інтерфейсу. Діаграма взаємозв'язків класів, що реалізують дану архітектуру для подання звітів ФОП 3 групи [7], подано на рис. 4.

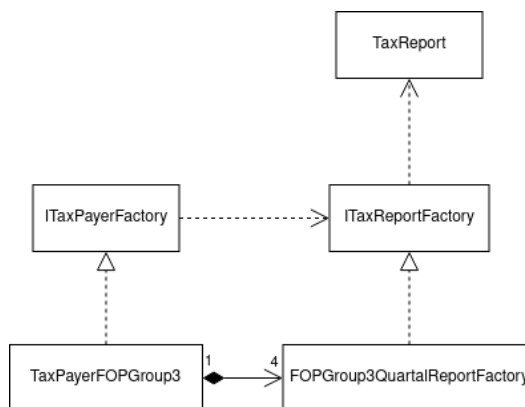


Рисунок 4. Діаграма взаємозв'язків класів, що реалізують створення звітів для ФОП 3 групи

3. ВИСНОВОК

Запропонована реалізація системи бухгалтерського обліку для ФОП як хмарної послуги дозволяє незалежне масштабування окремо модулів графічного інтерфейсу, бухгалтерських обчислень та бази даних. Це дозволяє ефективно використовувати наявні ресурси та адаптуватись до дійсної потреби користувачів під час виконання програми. Архітектура додатку дозволяє оновлювати його функціонал під час його роботи без порушення сесій

користувачів, поступово замінюючи екземпляри модуля бухгалтерських обчислень. Реалізація нових функцій вимагає зміни одночасно всіх модулів, що може мати більший час реалізації, але натомість існує ізоляція функціоналу додатку від певної мови програмування, що дозволяє використання інших типів графічних інтерфейсів (mobile, desktop). Поточний дизайн не передбачає використання модуля бухгалтерських розрахунків окремо від модуля графічного інтерфейсу, оскільки вони мають попередньо налаштоване з'єднання TLS, що захищає від втручання від зовнішніх агентів окрім серверу графічного інтерфейсу. Система може бути розширена за допомогою реалізації автоматичного подання податкової звітності до державної податкової служби України. Також можлива реалізація оплати розрахованої суми податків за допомогою існуючих систем оплати — GooglePay, MonoPay, тощо. Це розширення пропонується реалізовувати як окремі модулі, що масштабуються окремо від основних модулів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Streamlit • A faster way to build and share data apps. URL: <https://streamlit.io/> (дата звернення: 24.10.2023).
2. Nolan D., Lang D. Authentication for Web Services via OAuth. Springer. 2014. ISBN 978-1-4614-7900-0.
3. gRPC A high performance, open source universal RPC framework. *gRPC*. URL: <https://grpc.io/> (дата звернення: 24.10.2023).
4. Currier C. Protocol buffers. Springer. 2022. Mobile Forensics – The File Format Handbook. https://doi.org/10.1007/978-3-030-98467-0_9.
5. Kroc K., Kizun O., Skublewska-Paszkowska M. Performance analysis of relational databases MySQL, PostgreSQL, MariaDB and H2. Journal of Computer Sciences Institute. 2020. Vol. 14. P. 1–7. URL: <https://doi.org/10.35784/jcsi.1565> (date of access: 29.11.2023).
6. SQLAlchemy - the database toolkit for python. URL: <https://www.sqlalchemy.org/> (date of access: 24.10.2023).
7. Податковий Кодекс України : Закон України від 02.12.2010 р. № 2755-VI : станом на 23 листоп. 2023 р.

ОПТИМІЗАЦІЯ ПЕРЕДАЧІ ДАНИХ У БАГАТОКОРИСТУВАЦЬКИХ ВІДЕОІГРАХ

Сухарєв О.М.¹, Безносик О.Ю.²

Національний технічний університет України «Київський політехнічний інститут
імені Ігоря Сікорського», Київ, Україна

¹ suharev.alexei1@gmail.com, ² beznosyk.oleksandr@lil.kpi.ua [0000-0003-2775-6070]

Дослідження присвячено оптимізації передачі даних у багатокористувацьких відеоіграх. Результати дослідження вказують на те, що передача даних між клієнтом і сервером може бути досить об'ємною і вимагати значної пропускної спроможності, особливо зі збільшенням кількості гравців та об'єктів в ігровому світі. Новизною є більш технологічна оптимізації процесу передачі даних та покращення продуктивності мережевої частини гри.

Ключові слова: клієнт, сервер, багатокористувацький, оптимізація, мережа, онлайн-гра, відеогра.

1. ВСТУП

На сьогоднішній день, онлайн-ігри стали одним з основних прибутків для компаній, які займаються розробкою комп'ютерних ігор. Комерційні ігри повинні підтримувати розраховану на багато користувачів систему, а онлайн-ігрові платформи прагнуть забезпечити більш легке придбання цих продуктів, забезпечуючи підтримку все зростаючої аудиторії. У той же час, нові покоління ігрових консолей тісно пов'язані з привабливістю багатокористувацьких ігор.

Мережеві ігри вимагають як послідовності, так і чутливості мережевої взаємодії. Послідовність важлива для підтримки однакового набору даних для всіх гравців, у той час як зворотній зв'язок вимагає, щоб оновлення даних відбувалися якнайшвидше. Однак, ці дві вимоги часто суперечать одна одній, і вирішення цієї дихотомії між послідовністю та зворотнім зв'язком лежить в основі взаємодії в режимі реального часу [2].

Сучасні ігри, розраховані на багато користувачів, можуть мати десятки тисяч гравців з усього світу – це означає, що масштабованість обраної мережевої архітектури стає вкрай необхідною.

Крім того, онлайн-ігрові платформи прагнуть надати гравцям більш індивідуалізований контент відповідно до їхніх уподобань. Для цього є необхідним використання методів аналізу поведінки гравця, що також важливо для виявлення інших користувачів, які шахраюють або ведуть себе неадекватно в ігровому світі.

2. ЗНАЧЕННЯ ОПТИМІЗАЦІЇ ПЕРЕДАЧІ ДАНИХ

Оптимізація передачі – це комплексна тема, яка складається з таких пунктів як: стиснення даних, ефективні структури даних, асинхронна передача даних, “кешування” даних (збереження даних), пророцтво дій, оптимізація протоколу передачі даних.

Стиснення даних надають можливість зменшити об'єм даних, які передаються між клієнтом і сервером. Найпопулярніші алгоритми стиснення: “Gzip” та “LZ77”.

Ефективні структури даних розробляються для оптимізації продуктивності операцій додавання, видалення, пошуку та оновлення даних залежно від конкретних потреб програми. У контексті розрахованих на мережеві ігри, де існує високий рівень взаємодії та обміну даних

між клієнтами та сервером, ефективні структури даних відіграють критичну роль в оптимізації мережевої взаємодії та забезпеченні чуйності ігрового світу.

Асинхронна передача даних надає можливість більш ефективно використовувати мережеві ресурси.

Кешування даних запобігає повторній передачі даних, які вже були відправлені.

Пророцтво дій – можливість сервера пророкувати дії клієнта. Це може знизити необхідність частоті передачі даних для кожної дії.

Оптимізація протоколу передачі даних є процесом поліпшення ефективності та продуктивності мережевого протоколу, який забезпечує обмін інформацією між клієнтами та сервером у багатокористувацькому середовищі. Це важливий аспект розробки таких продуктів, і його метою є створення більш ефективного, швидкого і надійного механізму передачі даних.

3. ВПРОВАДЖЕННЯ ОПТИМІЗАЦІЇ ПЕРЕДАЧІ ДАНИХ

Оптимізована передача даних у розрахованих на багато користувачів іграх може значно вплинути на різні аспекти ігрового досвіду і продуктивності. Кілька областей, де можна очікувати різницю між оптимізованою та неоптимізованою передачею даних, наведено в Таблиці 1.

Загалом, оптимізована передача даних забезпечує більш стабільний, чуйний і плавний ігровий досвід, що позитивно впливає на задоволення гравців та загальну успішність самого ігрового продукту.

Таблиця 1. Критерії, що потребують оптимізації

Критерій	З оптимізацією	Без оптимізації
Затримки та зворотній зв'язок	Зменшення затримок при передачі даних може призвести до більш чутливого ігрового середовища. Гравці бачитимуть оновлення та реагуватимуть на події ближче до реального часу.	Затримки передачі даних можуть призвести до повільних оновлень, затримок у відображенні дій інших гравців та менш чутливої взаємодії.
Плавність ігрового процесу	Більш ефективна передача даних може забезпечити плавний рух об'єктів, точніше відображення стану ігрового світу та зменшення "лагів"(затримок) [4].	Можливі стрибки, затримки та невідповідності того, що бачать гравці у себе на екранах [4].
Пропускна здатність та надмірна навантаження	Зменшення обсягу даних, що передаються, зниження навантаження на мережу і забезпечення більш ефективного використання пропускної здатності.	Надмірне навантаження на мережу може призвести до втрати пакетів, затримок та зниження продуктивності гри.
Масштабованість	Гра з оптимізованою передачею даних буде легше масштабуватися зі зростанням кількості одночасних гравців.	Неоптимізована передача даних може призвести до проблем із продуктивністю за умови великої кількості гравців.
Ефективне використання ресурсів	Поліпшене використання ресурсів дозволяє грі працювати ефективно на різних пристроях та мережах [5].	Гра може мати проблеми з продуктивністю на слабких пристроях або в умовах обмеженої пропускної спроможності мережі [5].
Відсутність читерства (мухлювання)	Застосування ефективних заходів безпеки та перевірок на сервері дає можливість запобігати мухлюванню [4].	Неоптимізована передача даних може ускладнити реалізацію перевірок та відкрити ширші можливості для нечесної гри [4].

4. РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

4.1. Проблема нечесної гри

У будь-якій онлайн грі, шахраї не просто спрощують собі гру, а й погіршують чужий ігровий досвід. Є багато речей, які можна зробити, щоб запобігти нечесній грі. Але найголовніший принцип – дуже простий: не довіряй гравцеві. Завжди чекайте на гірше, а саме те, що гравець намагатиметься вас обдурити.

Авторитарний сервер та наївний клієнт:

Цей принцип веде нас до простого, на перший погляд, рішення – вся ігрова логіка крутиться на головному сервері, під вашим контролем, а клієнт лише демонструє поточний стан сервера та відправляє йому команди (натискання клавіш тощо). Зазвичай, це називають авторитарним сервером, тому що він єдиний, хто має право моделювати світ.

Наприклад:

Клієнт говорить, що я на позиції (10, 10), а через секунду – на позиції (20,10). При цьому можливо він «пройшов» через стіну або просто рухається швидше.

Правильна парадигма наступна: сервер знає, що гравець знаходиться в позиції (10, 10); клієнт каже: «Я хочу посунути на одиницю праворуч». Сервер оновлює позицію гравця на (11, 10), виконуючи всі необхідні перевірки, а потім відповідає гравцеві: «Ви на (11, 10)» (рис. 1).

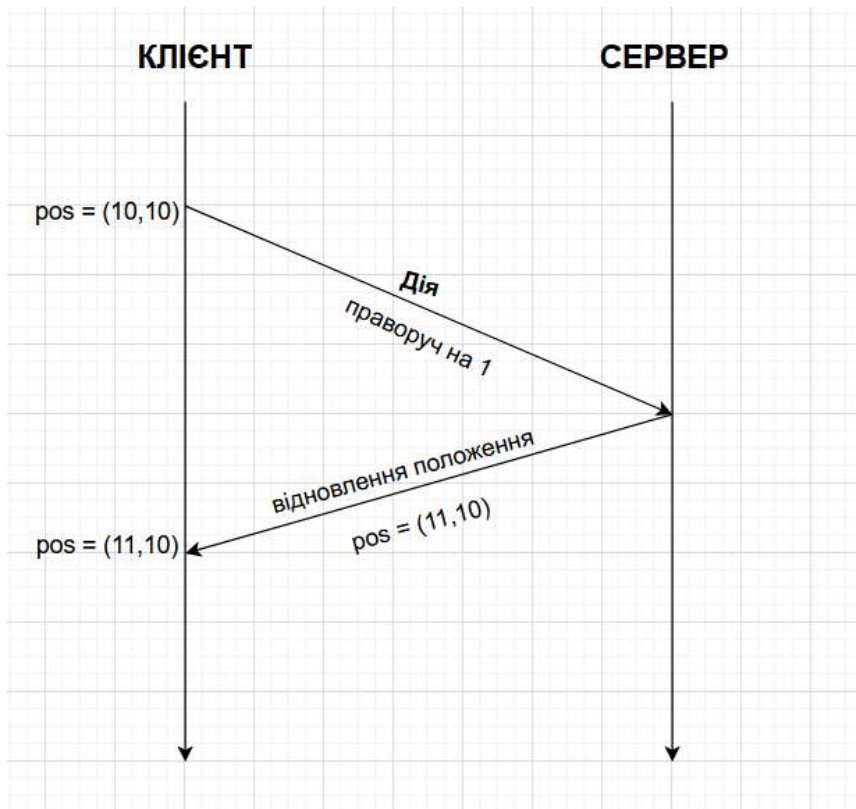


Рисунок 1. Приклад спілкування між клієнтом та сервером

Підсумок: ігровий стан керується лише сервером. Клієнти надсилають свої дії на сервер, а сервер періодично оновлює свій стан і відправляє його на клієнти, які, у свою чергу, відображають його користувачам.

4.2. Затримки

Наприклад, якщо гравець натискає стрілку вліво, персонаж почне рухатися через пів секунди. Це тому, що команда повинна дійти до сервера, а результат команди після цього повинен дійти до клієнта (рис. 2).

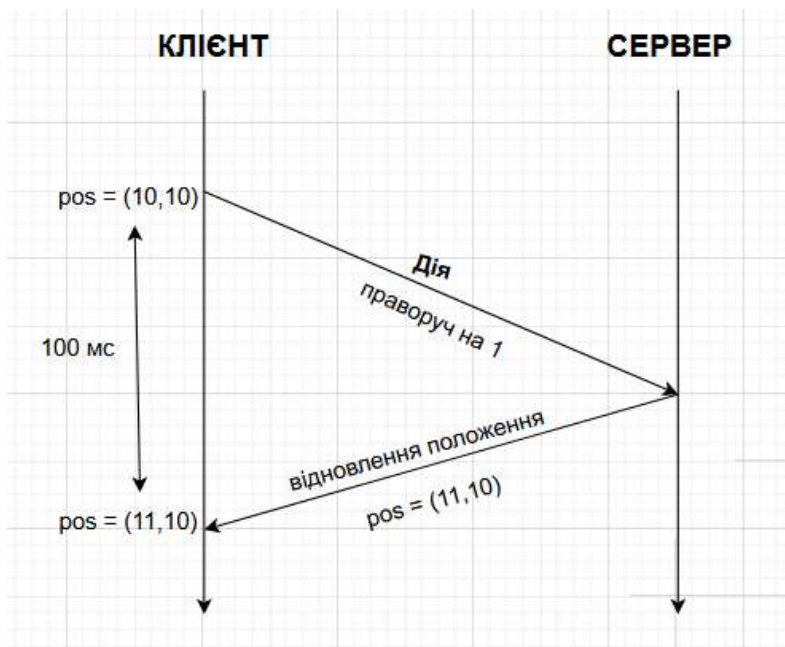


Рисунок 2. Затримка між командою та реакції на неї

В мережі, де затримки можуть становити десятки частки секунди, геймплей (ігровий процес) у кращому разі буде не чутливим.

Незважаючи на те, що деякі гравці намагаються вести нечесну гру, більшу частину часу сервер отримує коректні запити. Це означає, що отримане введення буде коректним і гра оновиться так, як очікується. Тобто, якщо персонаж знаходиться на позиції $(10, 10)$ і відправляє команду на рух праворуч, він опиниться на $(11, 10)$.

Ми можемо використовувати це якщо гра досить детермінована (тобто результат визначений командами та попереднім станом). Припустимо, що у нас лаг (затримка) 100 мс і час переміщення персонажа становить також 100 мс. При використанні наївної реалізації час дії складе 200 мс (рис. 3).

Замість того, щоб відправляти команду і чекати поки прийде новий ігровий стан, щоб отримати його зображення, ми можемо відправити команду і почати формувати зображення результату, якби команда вже була виконана. Тим не менш, треба чекати від сервера результату — «справжнього» стану гри, який здебільшого співпадатиме з локальним станом (рис. 4).

Тепер у нас немає абсолютно ніякої затримки між дією гравця та результатом на екрані, а сервер все ще авторитарний (якщо нечесний клієнт почне відправляти некоректні команди, це ніяк не вплине на стан гри на сервері, який бачать інші гравці).

Таким чином, при використанні авторитарного сервера, ви повинні надати гравцеві ілюзію чутливості, хоча насправді він чекає, поки сервер обробить введення. Для цього клієнт симулює результат усіх команд. Коли приходить оновлення від сервера, стан оновлюється залежно від поточного стану сервера та необроблених команд.

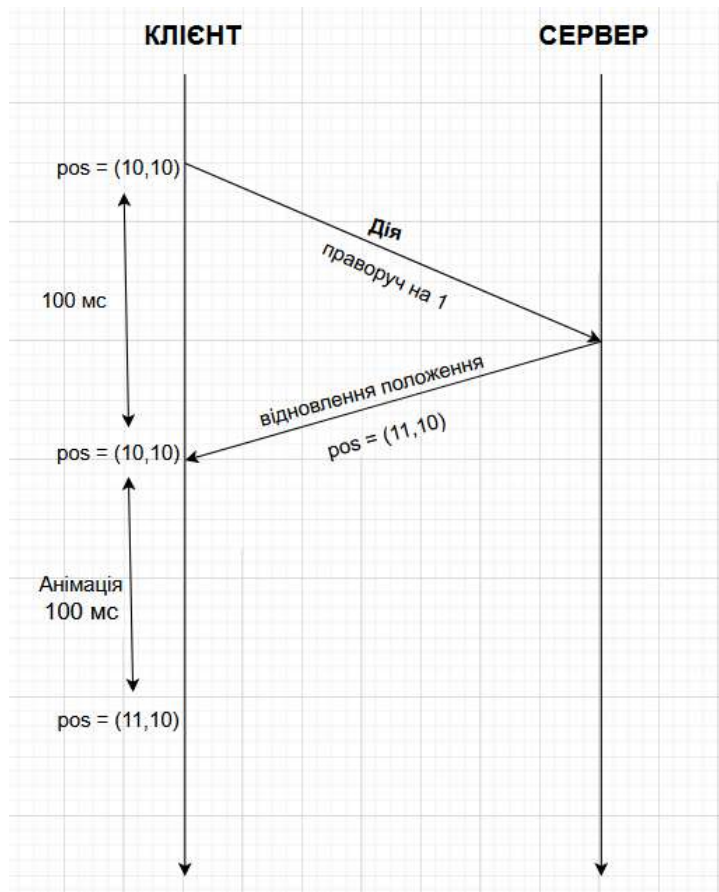


Рисунок 3. Приклад затримки реакції на дію користувача і затримка переміщення персонажа

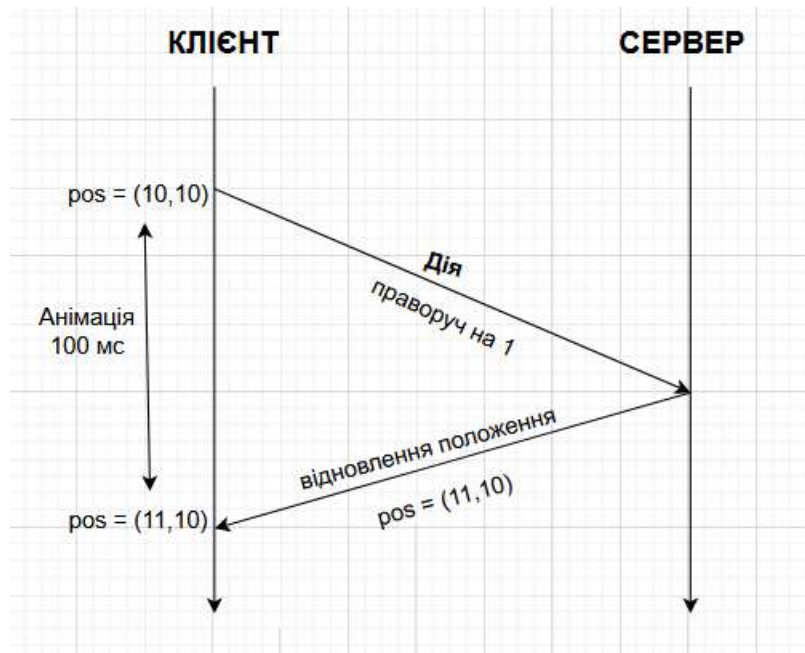


Рисунок 4. Формування зображення, не дочекавшись відповіді з сервера

5. ВИСНОВОК

У світі онлайн-ігор продуктивність ігрових серверів має першорядне значення для забезпечення безперебійного та приємного ігрового процесу. Зосередивши увагу на масштабованості, розміщенні сервера, ефективному мережевому зв'язку, апаратних ресурсах, оптимізації коду, заходах безпеки та моніторингу взаємодії з гравцями, можна створити ігрове середовище, яке буде підтримувати інтерес гравців та не викликати розчарувань.

Оптимізація продуктивності ігрового сервера не тільки покращує ігровий процес, але й сприяє формуванню лояльного та задоволеного співтовариства гравців, забезпечуючи довговічність та успіх онлайн-ігор у висококонкурентній галузі.

Дослідження підтверджує, що в онлайн-іграх, через високі вимоги до продуктивності мережі, навіть невеликі зниження продуктивності є дуже помітними і будуть вважатися недопустимими для кінцевого користувача.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Optimizing Game Server Performance for Seamless Gaming" URL: <https://medium.com/@furatamarizuki/optimizing-game-server-performance-for-seamless-gaming-experiences-540bbd8cb0f8>.
2. "Technology Interns in 2020: General Game Tech & Tooling" URL: <https://technology.riotgames.com/news/technology-interns-2020-general-game-tech-toolinginfrastructure>.
3. "How netcode works, and what makes 'good' netcode | PC Gamer." URL: <https://www.pcgamer.com/netcode-explained>.
4. "The DOOM III Network Architecture" URL: https://fabiensanglard.net/doom3_documentation/The-DOOM-III-Network-Architecture.pdf.
5. "Tips to Improve Game Networking for Online Games." URL: <https://www.linkedin.com/advice/0/how-do-you-improve-game-networking-skills-game-development>.

СИСТЕМА РОЗУМНОГО БУДИНКУ З КЕРУВАННЯМ ТЕЛЕГРАМ БОТОМ ТА GOOGLE ASSISTANT

Харченко К.В.¹, Кущовий Д.І.²

Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ, Україна

¹ konst1970@gmail.com, ² d.kuschevoy@gmail.com

Цей проект розробляє інноваційну систему розумного будинку, що інтегрується з Телеграм-ботом та Google Assistant для зручного та ефективного керування домашніми пристроями. Система дозволяє користувачам віддалено керувати освітленням, температурою, безпекою та іншими функціями будинку через прості голосові команди або текстові повідомлення. Особливу увагу приділено інтеграції з різними пристроями та платформами, а також забезпеченню безпеки даних та конфіденційності користувачів. Проект включає розробку інтуїтивно зрозумілого інтерфейсу, що робить управління будинком доступним і зрозумілим для широкого кола користувачів.

Ключові слова: розумний будинок, мікроконтролер, мікрокомп'ютер.

1. ВСТУП

Сучасний розумний будинок – це не просто набір окремих розумних пристроїв, а складна інтегрована система, що об'єднує контролери, датчики, різноманітні інтерфейси користувача та мережеві з'єднання. Всі ці елементи разом формують гармонійне середовище, яке реагує на потреби мешканців, адаптується до їхнього способу життя та надає нові можливості для керування домашнім простором [1].

2. ОСОБЛИВОСТІ РОЗУМНОГО БУДИНКУ

Особливості розумного будинку, розробленого з використанням Java framework Spring, що виступає у ролі сервера, та інтегрованого з мікроконтролерами ESP8266, демонструють високий рівень інновацій та технічної ефективності. Використання мікрокомп'ютера Raspberry Pi 4 як основи для Java сервісів ще більше підвищує гнучкість та функціональність системи [2].

2.1. Java Spring як сервер

Spring Framework є одним з найпопулярніших інструментів для розробки Java-додатків, зокрема завдяки його масштабованості та гнучкості. Використання Spring дозволяє легко інтегрувати різноманітні сервіси та компоненти, необхідні для функціонування розумного будинку.

Spring фреймворк забезпечує потужні засоби для забезпечення безпеки та надійності системи, що є важливим фактором для будь-якої системи розумного будинку.

2.2. Мікрокомп'ютер Raspberry Pi 4 для Java сервісів

Компактність та енергоефективність Raspberry Pi 4 забезпечується завдяки компактності, роблячи його ідеальним вибором для використання в якості серця системи розумного будинку.

Незважаючи на свої невеликі розміри, Raspberry Pi 4 забезпечує достатньо обчислювальної потужності для запуску Java сервісів, що вимагаються для управління розумним будинком.

2.3. Інтеграція з мікроконтролерами ESP8266

ESP8266 відомі своєю універсальністю та доступністю, що робить їх ідеальними для розробки різноманітних компонентів розумного будинку, таких як датчики та контролери.

Wi-Fi підключення відбувається завдяки вбудованому Wi-Fi модулю. ESP8266 можуть легко підключатися до мережі, що створюється Raspberry Pi 4 та Spring сервером, забезпечуючи надійну та швидку комунікацію.

3. ІНТЕГРАЦІЯ

Сервер, розроблений на основі Java Spring, відіграє ключову роль у мікросерверній архітектурі системи розумного будинку. Ця архітектура дозволяє інтегрувати нові клієнти та сервіси з високим рівнем гнучкості та ефективності.

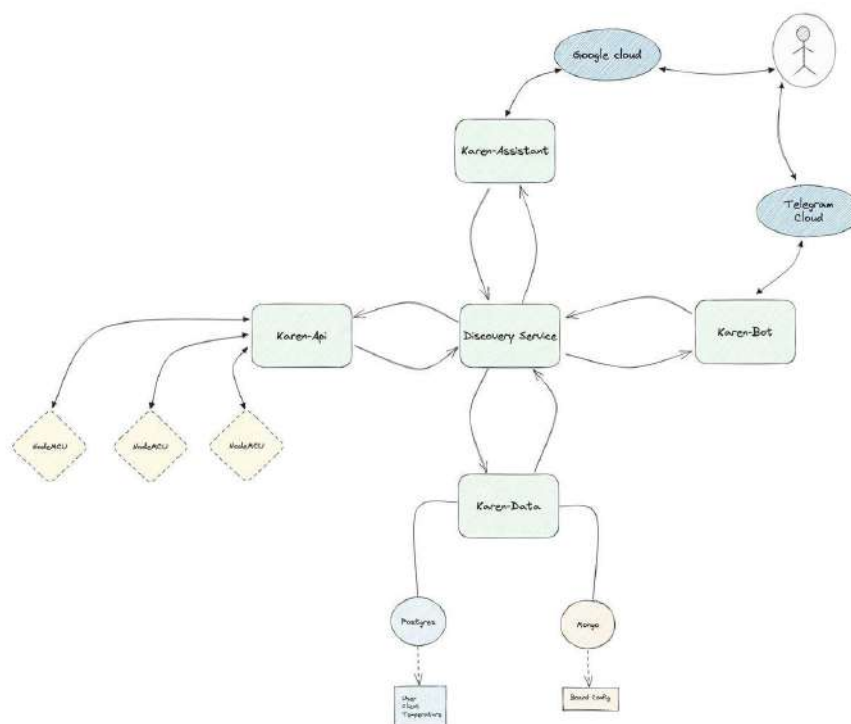


Рисунок 1. Схема архітектури системи

3.1. Інтеграція нового клієнта (мікроконтролера)

Інтеграція нового клієнта полягає у виконанні процесу запиту та реєстрації:

- клієнти, такі як мікроконтролери ESP8266, ініціюють інтеграцію, відправляючи HTTP-запити до сервера. Вони передають дані про свої можливості та управлінські функції;
- сервер на Java Spring обробляє ці запити, зберігаючи інформацію про клієнтів у базі даних. Це дозволяє системі ідентифікувати та управляти різними клієнтами ефективно.

Мікросерверна архітектура забезпечує легке підключення нових пристроїв, забезпечуючи високий рівень адаптивності системи без необхідності значних змін у її основній структурі.

3.2. Інтеграція нового сервісу

Процес запиту та отримання даних відбувається таким чином: при додаванні нового сервісу для графічного або голосового управління, цей сервіс ініціює зв'язок, відправляючи HTTP-запит до сервера. У відповідь, сервер надає інформацію про підключені клієнти, що дозволяє новому сервісу отримати доступ до управлінських функцій цих клієнтів.

Далі на основі отриманої інформації про клієнта з бази даних новий сервіс взаємодіє з сервісом-посередником для реалізації управління. Це забезпечує єдиний інтерфейс для різних типів сервісів, спрощуючи процес інтеграції.

4. ІНТЕРФЕЙСИ КОРИСТУВАЧА

4.1. Телеграм-бот

Використання Телеграм-бота для управління системою розумного будинку має ряд значних переваг. По-перше, Телеграм є одним із найпопулярніших месенджерів, який відрізняється високим рівнем безпеки та приватності, що є важливим аспектом при управлінні домашньою автоматизацією. Це означає, що команди та дані, які передаються через Телеграм-бота, захищені від несанкціонованого доступу.

Телеграм-бот також забезпечує високу доступність та зручність у використанні. Він дозволяє користувачам керувати своїм домом з будь-якої точки світу, де є доступ до інтернету, використовуючи звичний інтерфейс месенджера. Це робить управління розумним будинком простим і інтуїтивно зрозумілим, оскільки більшість користувачів уже знайомі з функціоналом Телеграму.

Ще однією перевагою є гнучкість у налаштуванні та кастомізації. Розробники можуть легко програмувати та налаштовувати бота для виконання специфічних завдань управління розумним будинком, від простого включення світла до більш складних дій, таких як моніторинг системи безпеки або регулювання температури [3].

Нарешті, інтеграція Телеграм-бота з системою розумного будинку може включати в себе розширений функціонал, такий як сповіщення про важливі події або зміни в стані системи, що забезпечує користувачам додатковий рівень інформованості та контролю над їхнім домом.

Таким чином, Телеграм-бот як інструмент для управління розумним будинком відкриває широкі можливості для зручності, безпеки та гнучкості, роблячи його ефективним та доступним рішенням для багатьох користувачів.

4.2. Google-assistant

Інтеграція Google Assistant у систему розумного будинку має низку значних переваг, які відкривають нові можливості для користувачів. Однією з основних переваг є зручність голосового управління. Користувачі можуть управляти різними функціями свого будинку, використовуючи прості голосові команди. Це не тільки економить час, але й робить управління більш інтуїтивним, особливо в ситуаціях, коли використання традиційних інтерфейсів незручне або неможливе.

Google Assistant також пропонує високий рівень персоналізації. Він може розпізнавати голоси різних користувачів і адаптувати реакції та дії згідно з їхніми індивідуальними налаштуваннями та звичками. Це створює більш персоналізоване та зручне середовище управління [4].

Крім того, Google Assistant відкриває можливості для розширеного інтелектуального взаємодії з домом. Він може надавати корисні сповіщення, нагадування та інформацію, пов'язану з побутовими завданнями та управлінням дому, наприклад, нагадування про

необхідність вимкнення пристроїв або зміну температури. Така взаємодія робить повсякденне життя більш комфортним та організованим.

Ще однією важливою перевагою є інтеграція Google Assistant з широким спектром пристроїв та сервісів. Це означає, що система розумного будинку може легко працювати в синергії з різноманітними побутовими пристроями та сервісами, що робить процес автоматизації більш гнучким та ефективним.

Отже, використання Google Assistant для управління розумним будинком значно підвищує зручність, персоналізацію та інтелектуальну взаємодію, що робить повсякденне життя більш комфортним та організованим [5].

5. ВИСНОВОК

З урахуванням викладеного вище, інтеграція як Телеграм-бота, так і Google Assistant в систему розумного будинку відкриває вражаючі можливості для покращення комфорту, безпеки та ефективності домашнього життя. Використання Телеграм-бота забезпечує безпечне, доступне та знайоме середовище для керування різними аспектами дому, пропонуючи простоту та гнучкість у використанні. Це, поєднуючи з його високим рівнем безпеки та приватності, робить Телеграм-бота відмінним вибором для інтерактивного управління системою розумного будинку.

З іншого боку, Google Assistant пропонує революційний підхід до управління домом за допомогою голосових команд, забезпечуючи високу ступінь персоналізації та інтуїтивно зрозумілого взаємодії. Він відкриває двері до більш інтелектуального і зв'язаного домашнього простору, де користувачі можуть легко взаємодіяти з своїми пристроями, отримувати сповіщення та управляти повсякденними завданнями з великою легкістю.

У поєднанні, ці технології не тільки підвищують зручність та ефективність управління домом, але й вносять значний вклад у створення адаптивного, реактивного та взаємопов'язаного домашнього середовища. Це відображає розвиток тенденцій у сфері домашньої автоматизації, де інноваційні технології відіграють ключову роль у перетворенні звичайного будинку на справді "розумний" дім.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. J. K. Bradley, "Smart Home Automation with Linux and Raspberry Pi", 2nd ed., Apress, 2013. ISBN 978-1430258872.
2. S. J. Weber, M. J. Wolf, "Internet of Things – IoT Systems: Architectures, Algorithms, Methodologies", Springer, 2018. doi: 10.1007/978-3-319-61300-0.
3. L. D. Paulson, "Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-learning", Morgan Kaufmann, 2008. ISBN 978-0123735942.
4. G. N. Preethichandra, "Wireless Sensor Networks for Developing Countries", Springer, 2013. doi: 10.1007/978-3-642-41013-0.
5. B. J. Bumgarner, "Programming Mobile Robots with Aria and Player: A Guide to C++ Object-Oriented Control", Springer, 2009. ISBN 978-1849962711.

ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ МІКРОСЕРВІСНИХ АРХІТЕКТУР ЧЕРЕЗ КЕШУВАННЯ ДАНИХ

Хоміч Л.І.¹, Яременко В.С.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

¹ lina.homich14@gmail.com, ² yaremenko.vadym@lil.kpi.ua

Дослідження спрямоване на оптимізацію продуктивності мікросервісних архітектур через впровадження ефективних стратегій кешування даних. Зосереджуючись на розгляді стратегій кешування та їх ефективності в умовах розподіленої системи, робота включає проведення аналізу. Під час дослідження визначається ефективність різних стратегій кешування в умовах розподіленої системи. Зазначається важливість вимірювання ефективності стратегій кешування для об'єктивного визначення їхнього впливу на продуктивність мікросервісів та виявлення оптимальних рішень для інформаційних систем.

Ключові слова: стратегії кешування даних, розподілені системи, CDN-мережі, продуктивність, оптимізація

1. ВСТУП

Сучасні технологічні трансформації висувають вимоги до найефективніших архітектур програмного забезпечення для забезпечення швидкості та надійності інформаційних сервісів. У цьому контексті, мікросервісні архітектури виявляються ключовим елементом, надаючи гнучкі та масштабовані рішення для технічних викликів.

Одним із аспектів оптимізації продуктивності мікросервісів є вдосконалення стратегій кешування даних. Кешування не лише прискорює відгук на користувацькі запити, але й ефективно розподіляє навантаження, покращуючи ефективність серверів та економлячи мережеві ресурси.

Враховуючи інтенсивність конкуренції та зростаючу різноманітність інтернет-сервісів, наше дослідження націлене на визначення оптимальних стратегій кешування. Шляхом аналізу та порівняння різних методів кешування, ми сподіваємося виділити науково-обґрунтовані та перспективні стратегії для забезпечення високої продуктивності інформаційних систем.

2. МІКРОСЕРВІСНІ АРХІТЕКТУРИ ТА СТРАТЕГІЇ КЕШУВАННЯ В УМОВАХ СУЧАСНИХ СИСТЕМ

2.1. Концепції розгортання мікросервісних архітектур.

Мікросервісні архітектури є передовою парадигмою розробки програмного забезпечення, де додаток розбивається на невеликі та автономні компоненти – мікросервіси. Розглянемо ключові концепції цього підходу та їхній вплив на продуктивність мікросервісних архітектур, особливо в контексті нашого дослідження.

Декомпозиція функціональності: Ця концепція передбачає розбиття функціональності додатку на невеликі, самостійні мікросервіси. Кожен з них відповідає за виконання конкретного завдання, сприяючи збереженню чистоти та структурованості коду.

Автономність та незалежність: Кожен мікросервіс працює відокремлено, дозволяючи розгортати, масштабувати та оновлювати його незалежно від інших компонентів системи. Це забезпечує гнучкість та ефективність у розробці та експлуатації.

Спрощена комунікація: Взаємодія між мікросервісами здійснюється за допомогою чітко визначених API, спрощуючи комунікаційні процеси. Це робить можливим ефективний обмін даними та реалізацію змін у функціональності без великих зусиль.

Гнучкість та масштабованість: Мікросервіси дозволяють використовувати різні технології для кожного компонента, що сприяє гнучкості розробки. Масштабованість окремих мікросервісів дозволяє ефективно використовувати ресурси в залежності від потреб системи.

Легкість утримання та розгортання: Завдяки своїй компактності та автономності, мікросервіси полегшують утримання системи та спрощують розгортання нових функціональних можливостей. Це особливо важливо у сучасному середовищі, де швидкість впровадження нових функцій має ключове значення.

CDN мережа з мікросервісами відкриває нові можливості для покращення продуктивності та доступності веб-додатків. Використання CDN для розгортання мікросервісів дозволяє оптимізувати розподілення контенту та обробку запитів, забезпечуючи ефективне використання ресурсів та знижуючи латентність для користувачів.

Хостинг мікросервісів у контексті CDN мереж надає новий рівень глобальної доступності та низької латентності для користувачів по всьому світу. Вибір відповідного CDN провайдера має велике значення для забезпечення оптимальної продуктивності та користувацького досвіду. Ефективне кешування та розподілення трафіку в межах CDN додають додаткові переваги управлінню хостингом мікросервісів, забезпечуючи стабільність та ефективність системи.

2.2. Стратегії кешування в умовах мікросервісів.

У випадку мікросервісних архітектур, що характеризується розподіленим характером та компактними розмірами сервісів, використання стратегій кешування даних виявляється ключовим елементом для забезпечення оптимальної продуктивності та швидкодії системи. Цей розділ присвячений аналізу різних стратегій кешування, їхнього впливу на роботу мікросервісних систем, а також визначенню оптимальних підходів для досягнення найкращих результатів у конкретних умовах.

Локальне кешування – одна з ключових стратегій, яку важливо розглянути. В цьому підході кожен мікросервіс зберігає власний кеш, що дозволяє зберігати часто використовувані дані прямо всередині мікросервісів. Такий підхід дозволяє мікросервісам ефективно отримувати доступ до необхідної інформації без постійного звертання до централізованих ресурсів. Кожен мікросервіс контролює свій кеш, сприяючи локалізації та забезпечуючи безпеку даних. Це не лише знижує часу доступу до даних, але й мінімізує навантаження на централізовані ресурси. Проте важливо враховувати, що зберігання окремих кешів у кожному мікросервісі може вести до дублювання інформації та надмірного використання простору. Цю стратегію можна кількісно характеризувати через такі характеристики: простір кешу – обсяг пам'яті для локального кешу визначається для кожного мікросервісу окремо; час доступу до даних - цей параметр визначає, наскільки швидко мікросервіс може отримати необхідні дані зі свого локального кешу. У порівнянні з іншими стратегіями, цей процес є швидшим, оскільки дані зберігаються безпосередньо у базі даних мікросервісу; дублювання даних – кількість дубльованих даних в локальних кешах визначається кількістю мікросервісів, які можуть зберігати однакові чи схожі дані в своїх кешах. Це може бути виміряно, наприклад, у відсотках загального обсягу даних в кешах або у кількості дубльованих записів.

Стратегія централізованого кешування передбачає використання спільного кешу, який накопичує дані для всіх мікросервісів. Цей підхід може бути ефективним у випадках, коли

декілька сервісів спільно використовують однакові дані. Завдяки централізованому кешу відбувається обмін та спільне використання даних між мікросервісами. Однак варто враховувати, що цей підхід може призвести до збільшення кількості запитів та обмеження гнучкості системи. Основними перевагами є спрощене управління, але це супроводжується можливістю зростання обсягу запитів та обмеженням гнучкості в архітектурі. Такий підхід зручний для сценаріїв, де великий обсяг даних має спільне використання, але його слабкі місця можуть стати проблемними в умовах збільшення обсягу запитів та потреби у вищій гнучкості системи. Оцінити дану стратегію можна як і попередню за обсягом спільного кешу та частотою обміну даними.

Розподілене кешування є важливою стратегією для підвищення продуктивності в мікросервісних архітектурах. В цьому підході кожен мікросервіс може мати свій власний кеш, розташований на різних вузлах мережі. Це сприяє швидкому доступу до даних, оскільки мікросервісам не потрібно постійно звертатися до централізованих ресурсів. Переваги розподіленого кешування включають швидкість доступу до даних, спрощене управління кешем для кожного мікросервісу та розподілення навантаження між різними частинами системи. Зокрема, такий підхід дозволяє системі ефективно працювати в умовах великої кількості розподілених компонентів. Однак розподілене кешування може стикається з викликами, такими як проблеми консистентності даних між різними кешами та додаткові складнощі у впровадженні та конфігурації. Незважаючи на це, враховуючи правильність конкретної ситуації, розподілене кешування може бути ефективним інструментом для покращення продуктивності мікросервісних архітектур. Дану стратегію можна характеризувати за кількістю вузлів кешу та швидкості обміну даними, що вимірюється як середня швидкість передачі даних між різними вузлами, і рівнем синхронізації між кешами, що показує наскільки консистентними та уніфікованими є дані між різними вузлами.

Інвалідація кешу та стратегії оновлення є важливим аспектом ефективного кешування в мікросервісних архітектурах. Ці стратегії визначають, коли і які дані слід оновлювати або видаляти з кешу для забезпечення актуальності інформації. Інвалідація кешу передбачає позначення певних даних як застарілих або невірних, що вимагає їх видалення або оновлення. Ця стратегія гарантує, що дані в кеші завжди відповідають актуальному стану системи. Проте необхідно правильно визначити моменти для проведення інвалідації, оскільки часте оновлення може впливати на продуктивність. Стратегії оновлення визначають частоту та способи оновлення даних в кеші. Для мікросервісних систем, де дані можуть змінюватися динамічно, важливо вибрати ефективний механізм оновлення. Враховуючи особливості кожної стратегії, важливо забезпечити збалансований підхід до інвалідації та оновлення кешу для оптимізації продуктивності [2]. Кількісною характеристикою даної стратегії є час інвалідації, а саме інтервал часу через який дані вважаються застарілими та підлягають інвалідації, впливає на актуальність інформації та навантаження системи під час оновлення. А також важливою характеристикою є спосіб оновлення даних в кеші: чи це періодичне оновлення, оновлення за запитом чи оновлення в реальному часі.

Управління життєвим циклом кешу в мікросервісних архітектурах є ключовою складовою для забезпечення оптимальної продуктивності системи. Ця стратегія включає в себе ряд методів, спрямованих на ефективне управління створенням, зберіганням і видаленням кешованих даних. Запобігання переповненню кешу та знищення застарілих даних – один із ключових аспектів управління життєвим циклом кешу. Вибір оптимальних методів зберігання кешу також важливий для управління його життєвим циклом. Це може включати в себе розробку ефективних алгоритмів очищення, які враховують активність та важливість даних. Керування тим, як довго дані залишаються в кеші, відіграє важливу роль у забезпеченні оптимальності використання ресурсів та збереженні потрібної інформації. Остаточо,

правильне управління життєвим циклом кешу визначає його ефективність та впливає на продуктивність мікросервісних архітектур в цілому [3]. Тому характеристикою даної стратегії виступає тривалість зберігання даних, а саме час, протягом якого дані залишаються в кеші до їх видалення або оновлення, впливає на доступність актуальних даних та ефективність кешування.

3. МОДЕЛЬ ТА СИМУЛЯЦІЯ МЕРЕЖІ

3.1. Розробка моделі CDN мережі

У даному розділі ми опишемо процес розробки моделі CDN-мережі, який включає в себе визначення параметрів, створення графової структури, імплементацію унікальних характеристик для кожного вузла, та побудову мережі для детального моделювання взаємодій між CDN провайдерами, серверами та кінцевими користувачами.

Першим кроком є визначення параметрів, які визначатимуть кожен вузол мережі. Це включає в себе:

Швидкість передачі даних: кожен сервер та CDN провайдер матиме свою швидкість передачі даних, що визначатиме ефективність обміну інформацією.

Обсяг пам'яті для кешування: величина, яка визначає, скільки даних може зберігатися в кеші кожного вузла.

Час TTL для збережених даних: час, протягом якого дані залишаються актуальними перед оновленням.

Мережу буде представлено у вигляді графу (Рис. 1), де вузли відобразатимуть сервери та CDN провайдерів, а листки графа представлятимуть кінцевих користувачів. Ребра графу відобразять швидкість передачі даних між вузлами.

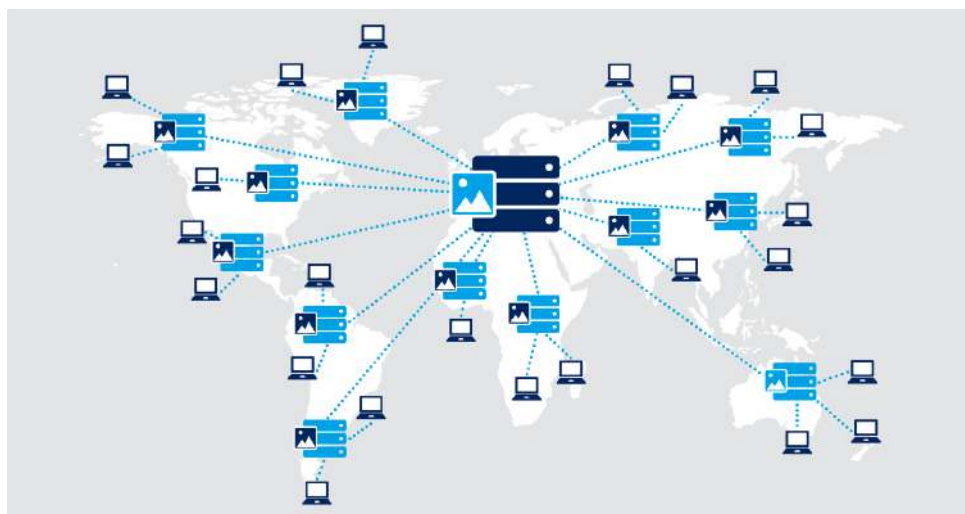


Рисунок 1. Схематичне представлення CDN мережі

Для кожного вузла, чи то сервер, мікросервіс чи CDN провайдер, будуть враховані унікальні характеристики, визначені під час визначення параметрів моделі. Це дозволить точно відтворити реальні умови використання CDN-мережі.

З урахуванням обраних стратегій кешування, реалізовані механізми взаємодії між вузлами мережі. Що включає відправлення запитів між мікросервісами, оновлення даних на головному сервері та поновлення кешованих даних.

Процес розробки моделі CDN-мережі є фундаментом для подальших симуляцій та експериментів з вибором стратегій кешування.

3.2. Вимірювання ефективності стратегій кешування

У цьому розділі ми розглянемо конкретні числові метрики та формули, які використовуються для вимірювання ефективності стратегій кешування в мережі CDN.

Час життя кешу (TTL) кешованих даних є критичним параметром, що визначає період, протягом якого дані можуть зберігатись в кеші перед їхньою інвалідацією. Ми вимірюємо TTL у секундах та позначаємо його як TTL_i , де i – індекс конкретного кешу.

Формула для обчислення часу інвалідації для даних на сервері S_j визначається як:

$$\text{Час інвалідації}_{ij} = \text{Час оновлення}_{S_j} + TTL_i.$$

Простір кешу позначає обсяг пам'яті, виділеної для зберігання кешованих даних. Для кожного мікросервісу S_j цей обсяг позначається як C_j (в байтах).

Час доступу до даних визначає, як швидко мікросервіс може отримати необхідні дані з свого кешу. Вимірюється в часових одиницях (зазвичай мілісекундах) і позначається як $T_{\text{доступу}}$.

Дублювання даних в локальних кешах визначається кількістю мікросервісів, які можуть зберігати однакові чи схожі дані в своїх кешах. Вимірюється відсотками загального обсягу даних в кешах або кількістю дубльованих записів.

$$\text{Дублювання}_{ij} = \frac{\text{Кількість дубльованих даних у кеші } S_j}{\text{Загальний обсяг даних у кеші}} \times 100\%.$$

Саме ці формули використовуються для вимірювання ефективності стратегій кешування.

4. ВИСНОВКИ

У цьому дослідженні ми проаналізували вплив різних стратегій кешування на продуктивність CDN-мереж у мікросервісних архітектурах. На основі нашої моделі CDN-мережі, представленої у вигляді графу, ми визначили та проаналізували стратегії, такі як локальне кешування, централізоване кешування, розподілене кешування, а також стратегії інвалідації та оновлення. Основні параметри нашої моделі включали швидкість передачі даних, обсяг пам'яті для кешування та час життя даних в кеші. Важливим аспектом дослідження стало висвітлення необхідності розроблення системи для моделювання середовища щоб вимірювати ефективності стратегій кешування. В даному симуляційному середовищі необхідно передбачити можливість задання критеріїв швидкість передачі даних, обсяг пам'яті для кешування та час життя даних в кеші, щоб отримувати оптимальну стратегію для кешування в умовах мікросервісів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pacheco V. F. Microservice Patterns and Best Practices / Vinicius Feitosa Pacheco. – 254 с
2. Barker T. Intelligent Caching / O'Reilly Media, Inc.- 107 с
3. Rabinovich, M., Spatscheck, O.: Web Caching and Replication. Addison Wesley - 52 с.

ГЕОЛОКАЦІЯ ПОЗИЦІЙ ОБ'ЄКТІВ РОБОЧОЇ КАРТИ ЗА ДАНИМИ ЗОБРАЖЕНЬ БПЛА

Хом'як К.В.¹, Петренко А.І.²

Національний технічний університет України «Київський політехнічний інститут
ім. Ігоря Сікорського», Київ, Україна

¹ karyna.khomiak@gmail.com, ² tolja.petrenko@gmail.com

Геолокація об'єктів на земній поверхні відіграє важливу роль у багатьох сферах, починаючи від аерофотограмметрії та геодезії й закінчуючи військовим застосуванням та навігацією. У сучасному світі безпілотні літальні апарати (БПЛА) стали одними з найбільш перспективних засобів для отримання геопросторової інформації через їхню здатність збирати великі обсяги даних відображення з висоти. Ці дані відкривають безліч можливостей, зокрема визначення місця розташування об'єктів на карті. Саме в цьому полягає необхідність в дослідженні геолокації позицій об'єктів, які б дозволили ефективно планувати і приймати подальші дії, виходячи з інформації просторового розташування важливих об'єктів, зібраних БПЛА. У цій статті підкреслюється важливість використання дронів та розглядається розробка архітектури системи передачі та аналізу даних, отриманих з БПЛА, для використання у військових цілях.

Ключові слова: БПЛА, дрони, безпілотники, геолокація, застосунок, військова галузь.

1. ВСТУП

Геолокація – це корисний інструмент для визначення географічного положення об'єкта чи системи за допомогою конкретних координат планети. Пристрої GPS використовуються в різних морських функціях, картографуванні, програмах для громадського використання. Також широко використовуються в робототехніці для перевірки, виявлення, спостереження, локалізації та завдань для автономних навігаційних систем. Безпілотні літальні апарати (БПЛА) є найпопулярнішими транспортними засобами для розробки завдань навігації за допомогою пристроїв GPS шляхом отримання місцезнаходження БПЛА за допомогою супутникової інформації під час навігації безпілотника [1].

БПЛА, також відомі як дрони, отримали важливе значення в різних дисциплінах військової та цивільної служби завдяки своїй підвищеній стабільності та витривалості в кількох операціях. Застосування БПЛА надзвичайно розширюється завдяки їх прогресивному використанню в Інтернеті речей (IoT), 5G. За останнє десятиліття БПЛА використовувалися в різних сферах застосування, включаючи виявлення та відстеження об'єктів, громадську безпеку, спостереження за дорожнім рухом, військові дії, дослідження прихованих або небезпечних зон, навігацію в приміщенні або на відкритому повітрі, зондування атмосфери, операції після катастроф, охорону здоров'я, обмін даними, управління інфраструктурою, управління надзвичайними ситуаціями та кризовими ситуаціями, вантажні перевезення, моніторинг лісових пожеж та логістика [2].

Безпілотні літальні апарати можуть виконувати повітряні операції, які пілотовані літальні апарати вважають складними. Використання безпілотних літальних апаратів

забезпечує значну економію та користь для навколишнього середовища, водночас зменшує ризики для життя людини. БПЛА здатний літати та залишатися в повітрі, не вимагаючи жодного оператора на борту, забезпечуючи економніші операції, ніж еквівалентні пілотовані системи, і виконує економічно ефективну критичну місію без ризику для життя людини. Ним можна дистанційно керувати, за допомогою чого команди управління надходять із наземної базової станції (БС) через пульт дистанційного керування. Вони також здатні виконувати контрольні операції на борту за допомогою автопілота та різних датчиків, включаючи глобальну систему позиціонування (GPS) та інерціальні вимірювальні пристрої (IMU) [3].

Зокрема, БПЛА привернули значну увагу провідних інтернет-магазинів, включаючи Walmart, DHL, Google і Amazon [4, 5].

Індустрія безпілотників привертає значну увагу як модель конвергенції виробництва, обслуговування та доставки, запроваджуючи синергію з співіснуванням різних нових сфер. БПЛА пропонують неявні особливості, такі як збільшений час у повітрі та можливості корисного навантаження, швидку мобільність та доступ до віддалених районів і зон стихійного лиха [3].

Особливий акцент в дослідженні робиться на військовій галузі, адже саме в цій галузі дрони відіграють критичну роль у зборі розвідувальної та стратегічної інформації. Вони забезпечують можливість нагляду, розвідки, слідкування за діями противника та збору даних без високого ризику для життя військовослужбовців. Дрони стають важливим інструментом для військових операцій, сприяючи покращенню стратегічного аналізу та виконанню завдань у складних умовах. Використання дронів у військовій галузі забезпечує доступ до важливої інформації, підвищує продуктивність та допомагає зменшити ризики для людей, забезпечуючи нові можливості для ефективного використання даних.

У цій статті розглядається архітектура комплексної програми, спрямованої на складання робочої карти для військових за допомогою даних, отриманих від БПЛА. Така програма допоможе військовому командуванню приймати оптимальні рішення щодо ходу бойових дій на основі даних, які вони переглядатимуть зі свого робочого місця.

2. АРХІТЕКТУРА КОМПЛЕКСНОЇ ПРОГРАМИ

2.1. Вимоги до програми

Ціль дослідження – розробити програму, яку одразу можна буде передати для використання військовими та тестуванням її у дії.

Волонтерські ІТ-центри допомоги військовим зазначили рекомендації, яких потрібно дотримуватися, щоб досягти максимально користі та практичності використання програми:

- програма має бути представлена у вигляді системи, розділеної на застосунки, відповідно до використання її звичайними військовими та вищим командуванням;
- згідно практичності застосунок для звичайних військових має бути представленим у вигляді мобільного додатку, яким вони будуть передавати дані;
- згідно практичності застосунку для вищого командування – це має бути комп'ютерний застосунок, з якого зручно переглядати інформацію та складати подальші прогнози;
- військові, що передають дані, мають змогу писати власні коментарі до зображень.

Також до вимог системи додається:

- зручний інтерфейс взаємодії користувача із застосунками;
- відсутність зайвих функцій та максимальна простота;
- можливість легкого масштабування системи для додавання необхідних функцій, коли такі виникнуть у процесі користування системою.

2.2. Узагальнений опис архітектури системи

Згідно вимог розроблено архітектуру системи, що включає в себе три основні частини та забезпечує їх взаємодію у використанні різними користувачами. Ці частини спільно працюватимуть для забезпечення ефективності та точності процесу нанесення геолокації на робочу карту.

Перша складова системи – мобільний застосунок. Ця частина призначена для збору та передачі фотографій з геомітками, отриманих від БПЛА, для подальшої обробки. Військовий має змогу зі свого кабінету завантажувати фотографії та вказувати додаткову інформацію.

Друга частина системи – сервісна частина, яка буде відповідальною за обробку та аналіз даних, що надходять від мобільного застосунку.

Третя частина системи – комп'ютерний застосунок, що спрямований на аналіз інформації та фотографій вищим командуванням. Цей інструмент надасть можливість для аналізу та висновків на основі отриманих даних, спрощуючи процес вирішення завдань та прийняття рішень. Вище командування матиме змогу переглядати карту України, зокрема її частину зони бойових дій – так звану робочу карту, на якій буде нанесено усі геомітки з інформацією. Відповідно користувач зможе за заданими координатами переглядати усі наявні фотографії у певному радіусі або через галерею відповідно в порядку останніх завантажень.

Ця комплексна система має на меті створення ефективного та надійного інструментарію для аналізу геолокації об'єктів на основі зображень, забезпечуючи можливість швидкого прийняття рішень на основі аналізу цих отриманих даних.

2.3. Опис мобільного застосунку

Мобільний застосунок, що призначений для використання звичайними військовими і передачі дані на сервер, має інтерфейс, показаний на рис. 1:

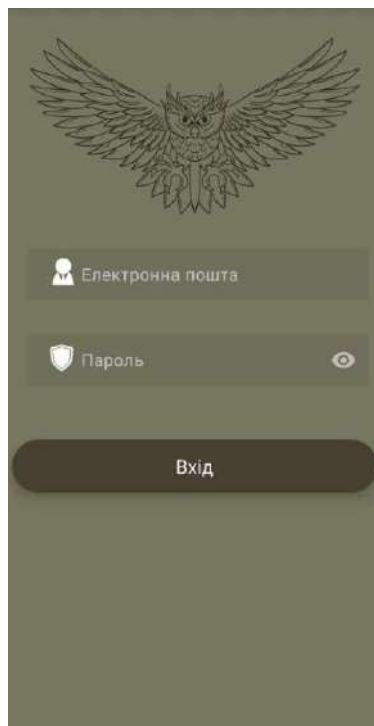


Рисунок 1. Інтерфейс входу до застосунку

Після того, як користувач увійшов в застосунок, він матиме змогу надати застосунку дозвіл доступ до галереї смартфона та завантажити необхідні фото з даними.

На рис.2 приведено вигляд вкладки із щойно завантаженим фото, де одразу видно основні дані, такі як саме фото, геолокація та коментар. Також є можливість додати фото із самої камери смартфона та задіяти кнопку відправки фотознімка на сервер.



Рисунок 2. Відображення сторінки із завантаженим фото

3. Додати коментар до фото користувач зможе також через текстове поле, показане на рис.

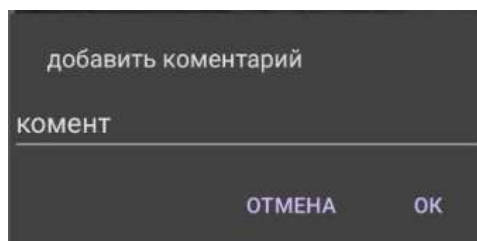


Рисунок 3. Вкладка «Додати коментар»

2.4. Опис сервісної частини

На сервері маємо базу даних, що зберігає такі дані:

- довгота;
- широта;
- висота;
- коментар;
- час знімку;
- час завантаження фото;
- самий фотознімок.

2.5. Опис комп'ютерного застосунку

Комп'ютерний застосунок відкриватиметься на головній вкладці «Мапа» та відображатиме карту України, яку буде змога масштабувати відносно будь-якої точки чи зони, що цікавить користувача (рис. 4). При ставленні маркера відбуватиметься пошук фотографій, що знаходяться найближче до зазначеного маркера та відображатимуться праворуч від мапи разом з коментарями, тобто будуть відображені останні завантажені знімки для цієї місцевості.

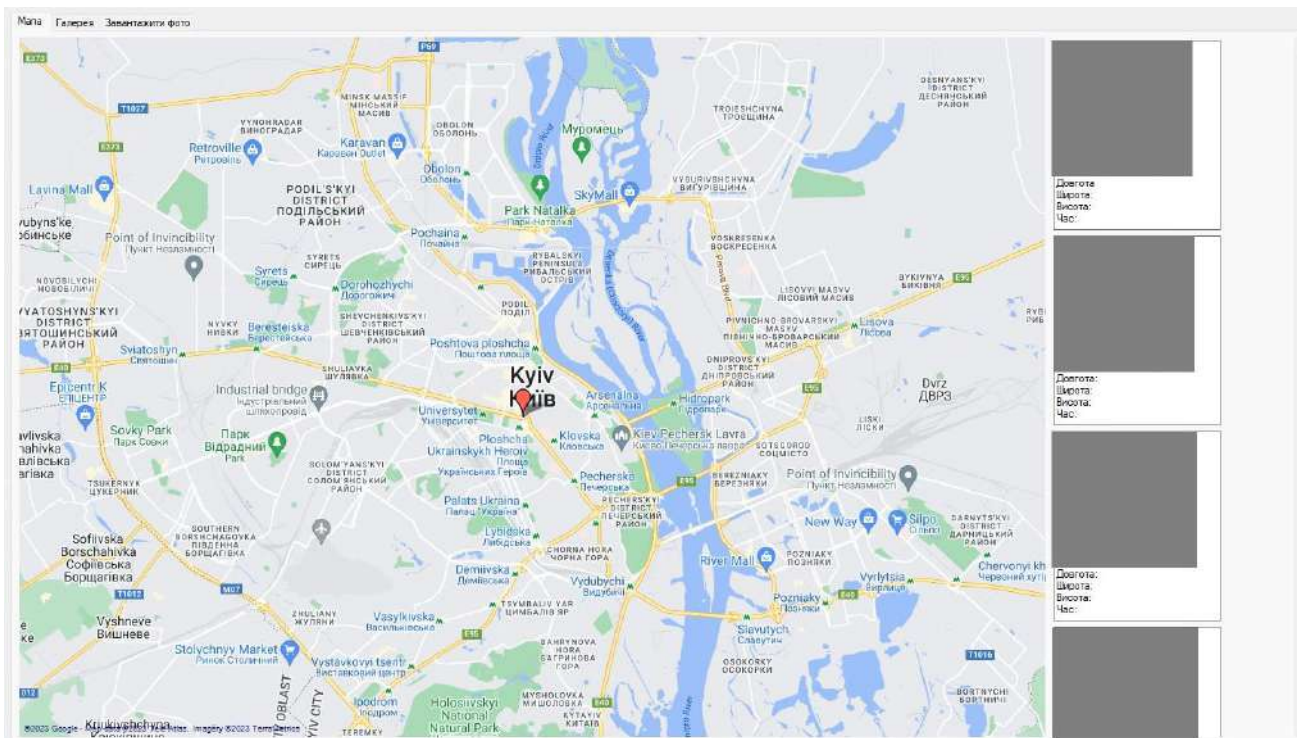


Рисунок 4. Схематична мапа України з міткою

У вкладці «Завантажити фото» маємо дві кнопки «Відкрити», що обирає фото на комп'ютері, та «Відправити», яка завантажує фото до системи. Також можемо перетягнути фото в виділену область для того, щоб завантажити файли.

У завантажених фото, що бачимо на вкладці до того, як їх буде відправлено до системи, маємо змогу переглядати важливі дані, а саме (рис. 5):

- фото;
- довготу;
- широту;
- висоту;
- час (із датою).

Маємо змогу видалити зайві фото, до того як їх буде відправлено.

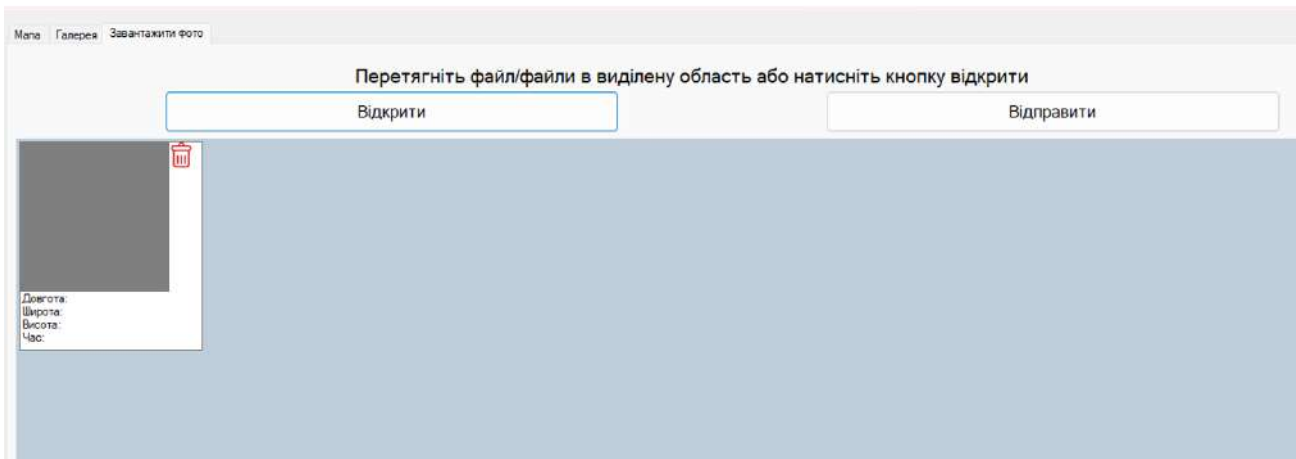


Рисунок 5. Вигляд вікна завантаження фото

У вкладці «Галерея» (рис. 6) будуть висвітлені останні за датою завантаження фото з даними та коментарями, що знаходяться знизу під фото відповідно до певних значень широти і довготи, а також діапазону часу, протягом якого буде йти пошук.

У даній вкладці реалізується також функція корекції по координаті.

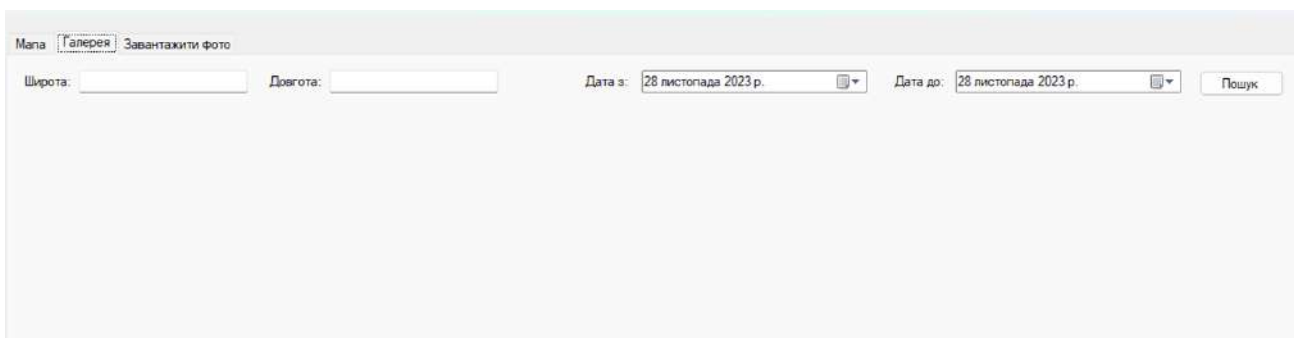


Рисунок 6. Вигляд вкладки «Галерея»

3. ВИСНОВКИ

Розглянута архітектура системи, яка складається з мобільного застосунку для військових, серверної частини та комп'ютерного застосунку для вищого командування, що є важливим кроком у розвитку засобів практичного використання геолокаційних даних.

Перш за все, ця система демонструє інтеграцію технологій у зборі, обробці та аналізі геоданих. Вона дозволяє військовим здійснювати збір та передачу фотографій з геомітками за допомогою мобільного застосунку, що відображає її практичність та зручність у використанні.

Крім того, серверна частина відповідає за обробку та аналіз отриманих даних, що вказує на можливість виконання складних операцій з геоданими у реальному часі.

Найбільш важливою складовою цієї архітектури є додаток для вищого командування, який надає можливість аналізувати робочу карту з геомітками. Це створює можливість для прийняття обґрунтованих рішень на основі даних та сприяє ефективному керуванню в умовах бойових дій.

У цілому, враховуючи можливості кожної складової системи, можна зробити висновок про їхню сумісність та спрямованість на створення ефективного та надійного інструментарію для аналізу геолокаційних даних із фото, сприяючи прийняттю оперативних та обґрунтованих рішень в умовах нестабільності та напруженості ситуацій.

У контексті дослідження геолокації позицій об'єктів за даними зображень БПЛА, розглянута архітектура системи є ключовою для забезпечення ефективного збору, передачі та аналізу цих важливих даних. В статті відзначено важливість безпілотних літальних апаратів як джерела великих обсягів геопросторової інформації, що відкриває широкі можливості для визначення геолокації об'єктів для потреб військової сфери.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “Geo-localisation of aerial images captured with drones” Awarded to Aldrich A. Cabrera-Ponce, MSc student supervised by Dr. Jose Martinez-Carranza, 2019.
2. Hassija, V.; Saxena, V.; Chamola, V. Scheduling drone charging for multi-drone network based on consensus time-stamp and game theory. *Comput. Commun.* 2019, 149, 51–61. URL: <https://www.sciencedirect.com/science/article/abs/pii/S014036641930948X?via%3Dihub> (дата звернення: 26.10.2023).
3. Mohsan, S.A.H.; Khan, M.A.; Noor, F.; Ullah, I.; Alsharif, M.H. Towards the Unmanned Aerial Vehicles (UAVs): A Comprehensive Review. *Drones* 2022, 6, 147. URL: <https://www.mdpi.com/2504-446X/6/6/147> (дата звернення: 30.10.2023).
4. Yoo, W.; Yu, E.; Jung, J. Drone delivery: Factors affecting the public’s attitude and intention to adopt. *Telemat. Inform.* 2018, 35, 1687–1700. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0736585318300388?via%3Dihub> (дата звернення: 15.11.2023).
5. Macrina, G.; Pugliese, L.D.P.; Guerriero, F.; Laporte, G. Drone-aided routing: A literature review. *Transp. Res. Part C Emerg. Technol.* 2020, 120, 102762. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0968090X20306744?via%3Dihub> (дата звернення: 15.11.2023).

САМОНАВЧАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ ЛЮДСЬКОЇ ДІЯЛЬНОСТІ З ВИКОРИСТАННЯМ НАВЧАННЯ З ПІДКРІПЛЕННЯМ

Цибін М.Д.¹, Кислий Р.В.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

¹ accessmaksym@gmail.com, ² kvrware@gmail.com

У даній роботі метою було дослідити та покращити адаптивність систем розпізнавання людської діяльності (HAR) під конкретного користувача, враховуючи його особливості рухів та зміни рухової поведінки з часом за допомогою методів Reinforcement Learning. Дослідження проводилося з використанням базових моделей LSTM, LGBM у симбіозі з навчанням з підкріпленням методами Монте Карло. Новизна і практична значимість визначається змогою системи автоматично коригувати моделі залежно від змін поведінки користувача, за допомогою навчання з підкріпленням.

Ключові слова: розпізнавання людської діяльності, навчання з підкріпленням, машинне навчання, штучний інтелект

1. ВСТУП

Системи розпізнавання людської діяльності можуть використовуватися для багатьох цілей, зокрема сферами використання можуть бути: спорт, медицина, контроль робочих процесів на виробництві, визначення розподілу часу у щоденній рутині, і якщо подумати, то досить багато аспектів застосування можна визначити для таких систем. У дослідженні акцентувалося саме на визначенні спортивних та щоденних активностей, таких як біг, ходьба, підйом сходами вгору та вниз, стояння та лежання. Початковий набір даних складався з даних датчиків по 3 осям акселерометра і гіроскопа на мобільному пристрої. На виході було за мету зробити систему з використання навчання з підкріпленням, яка б мала змогу автоматично адаптуватися до конкретного користувача та під зміни його рухової поведінки з часом.

2. БАЗОВІ МОДЕЛІ (БЕНЧМАРК)

Як базові моделі було обрано LSTM та LGBM. Ці моделі добре підходять для розпізнавання людської діяльності, що є задачею класифікації. Моделі було протестовано на початковому наборі даних, отримана точність прогнозу 0.9455 для LSTM та 0.9845 для LGBM. Це гарний результат, що повністю задовольняє нашу ціль.

LSTM

LSTM (Long Short-Term Memory) — це тип рекурентних нейронних мереж, який добре підходить для моделювання послідовностей та часових рядів. У дослідженні з розпізнавання людської діяльності LSTM має купу переваг, адже мова йде про аналіз часових залежностей у даних.

LSTM є вдосконаленням типової рекурентної нейронної мережі (RNN), яка має здатність зберігати та використовувати інформацію на тривалий термін, що робить його корисним для аналізу довгих послідовностей, де важлива контекстна залежність.

Основна ідея полягає в тому, що LSTM має механізми видалення та додавання інформації в пам'ять на різних часових шарах. Він складається зі спеціальних "ворот", таких як ворота забування (forget gate), ворота входу (input gate), вихідні ворота (output gate). Ці ворота дозволяють регулювати потік інформації в внутрішнім станом моделі.

Форгет-ворота визначають, яка інформація буде забута чи збережена у пам'яті моделі. Ворота входу визначають, яка нова інформація буде додана до пам'яті. Вихідні ворота визначають, яка інформація буде використана для прогнозування наступного значення.

Ця архітектура дозволяє LSTM ефективно працювати з послідовностями різної довжини, уникати проблеми зникаючого/вибухаючого градієнта, яка властива звичайним рекурентним мережам.

LGBM

Light GBM (Light Gradient Boosting Machine) – це потужний алгоритм машинного навчання, який належить до сімейства ансамблевих методів, зокрема, до методу градієнтного підсилення (Gradient Boosting). Він став досить популярним завдяки своїй ефективності та швидкодії, особливо при обробці великих обсягів даних.

Light GBM використовує техніку підсилення градієнта, де модель навчається ітеративно, додаючи нові базові моделі (зазвичай дерева рішень) для виправлення помилок попередніх моделей. Основна ідея полягає в тому, щоб кожна нова модель удосконалювала попередні, фокусуючись на невирішених підсумкових помилках.

Однією з основних переваг Light GBM є його швидкодія. Він використовує градієнтний спуск для знаходження оптимальних значень параметрів моделі, а також оптимізує процес побудови дерев, використовуючи методи, які дозволяють ефективно працювати з великими обсягами даних.

3. НАВЧАННЯ З ПІДКРІПЛЕННЯМ

МОНТЕ КАРЛО

Monte Carlo Reinforcement Learning – це підхід до машинного навчання, який використовує ідеї з області теорії імітаційного моделювання та навчання з підкріпленням для прийняття оптимальних рішень в умовах невизначеності.

Основна ідея методу полягає в тому, що агент навчається шляхом випадкового (монте-карло) взяття вибірки з можливих дій у певному середовищі і спостереження результату цих дій. Агент збирає досвід про те, які дії призводять до найкращих результатів в певних ситуаціях.

Важливо враховувати, що метод монте-карло не завжди є ефективним в умовах великої кількості можливих станів або дій, оскільки вимагає значної кількості випробувань для знаходження оптимального рішення. Однак він може бути корисним для моделювання навчання людини або агента в умовах невизначеності та відсутності чітких правил.

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
 $Q(s, a) \leftarrow$ arbitrary
 $Returns(s, a) \leftarrow$ empty list
 $\pi \leftarrow$ an arbitrary ϵ -soft policy

Repeat forever:
 (a) Generate an episode using π
 (b) For each pair s, a appearing in the episode:
 $R \leftarrow$ return following the first occurrence of s, a
 Append R to $Returns(s, a)$
 $Q(s, a) \leftarrow$ average($Returns(s, a)$)
 (c) For each s in the episode:
 $a^* \leftarrow \arg \max_a Q(s, a)$
 For all $a \in \mathcal{A}(s)$:
 $\pi(s, a) \leftarrow \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \epsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$

Рисунок 1. Псевдокод методу Монте Карло

4. СИМБІОЗ МОДЕЛЕЙ

У розробленій системі (Рис. 2) моделі LSTM і навчання з підкріпленням методом Монте Карло поєднуються для досягнення більшої ефективності. Модель метода Монте Карло на основі тренувальних даних генерує нові дані схожі на попередні але вони мають більш виразні ознаки активностей. Далі ці дані приймає на вхід наша базова модель, до прикладу LSTM. модель тренується і готова приймати подальші активності на вхід для визначення прогнозу стану. З часом коли типова поведінка користувача змінюється і модель деградує, за критеріями точності прогнозу, модель перетреноується автоматично на нових даних адже система зберігає дані попередніх активностей для майбутнього покращення моделі. Такий симбіоз у системі дозволяє використовувати плюси обох моделей і робити процес прогнозування для користувача більш точним та адаптивним.

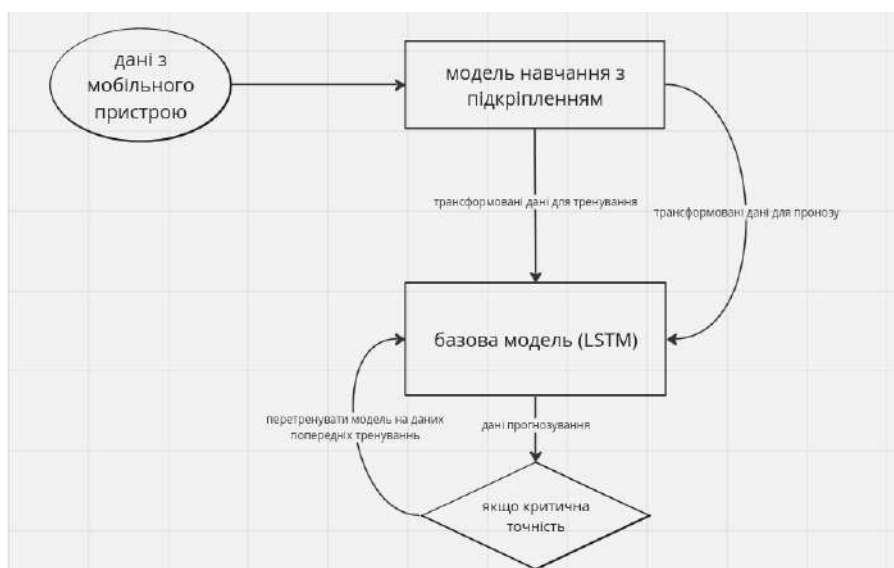


Рисунок 2. Схема роботи системи

5. РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

Були виконані дослідження з тренування моделей на датчиках з мобільного пристрою, використовувалися дані по 3 осям гіроскопа та акселерометра. Наведемо дані з тренування та тестування нашої системи з моделей LSTM та навчання з підкріпленням метода Монте Карло.

Модель це наша LSTM, що є вдосконаленням RNN моделі. Вона складається з 5 шарів та має 79302 параметри що тренуються (Рис. 3).

Layer (type)	Output Shape	Param #
input_17 (InputLayer)	[(None, 128, 9)]	0
lstm_16 (LSTM)	(None, 128, 128)	70656
dense_32 (Dense)	(None, 128, 64)	8256
global_max_pooling1d_16 (GlobalMaxPooling1D)	(None, 64)	0
dense_33 (Dense)	(None, 6)	390
...		
Trainable params: 79302 (309.77 KB)		
Non-trainable params: 0 (0.00 Byte)		

Рисунок 3. Характеристика LSTM моделі

Наведемо Loss графік (Рис. 4), який відображає зміну значення функції втрати (loss function) під час процесу навчання моделі машинного навчання. Функція втрати визначає, наскільки добре модель працює на даному наборі даних під час навчання, порівняно зі справжніми (очікуваними) значеннями.

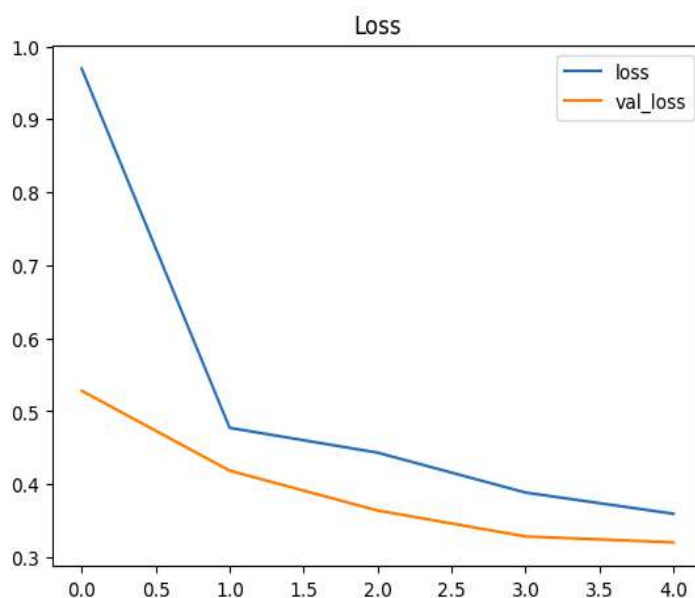


Рисунок 4. Графік зміни функції втрати при тренуванні моделі

Також наведемо графік точності (ассигасу), який відображає зміну точності моделі під час процесу навчання (Рис. 5). Точність – це метрика, яка визначає, наскільки точно модель класифікує дані. Вона вимірюється у відсотках та показує, який відсоток прикладів модель класифікувала правильно відносно загальної кількості прикладів.

Під час кожної ітерації або епохи навчання моделі використовуються навчальні дані для оновлення параметрів моделі. Після цього використовується окремий набір валідаційних або тестових даних, для оцінки точності моделі на даних, які вона не бачила під час навчання.

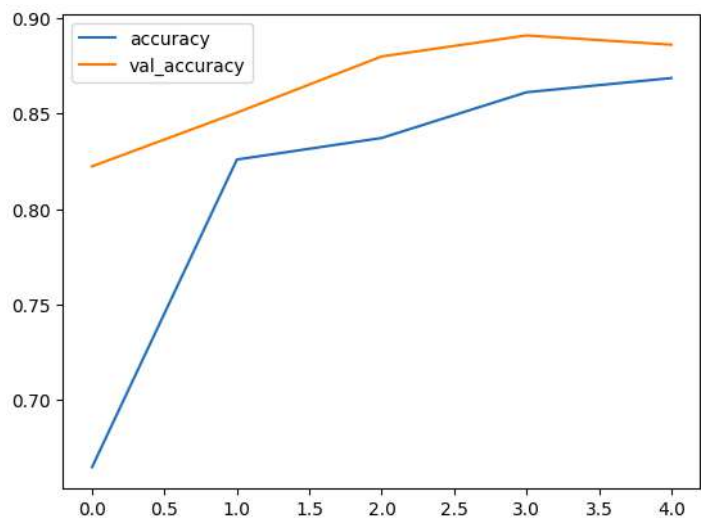


Рисунок 5. Графік росту точності при тренуванні моделі

Також ми побудували матрицю невідповідностей (Confusion Matrix) – це таблиця, яка дозволяє візуалізувати результати класифікації моделі на тестовому наборі даних, де відображається кількість правильних і неправильних прогнозів для кожного класу (Рис. 6).

Кожен рядок у матриці невідповідностей відповідає фактичному класу, а кожний стовпчик – передбаченому класу. Основна діагональ цієї матриці відображає правильно класифіковані приклади (коли фактичний клас співпадає з передбаченим). Елементи поза діагоналлю представляють помилки у класифікації: наприклад, якщо вибіркового класифікатора помилково визначив біг як ходьбу.

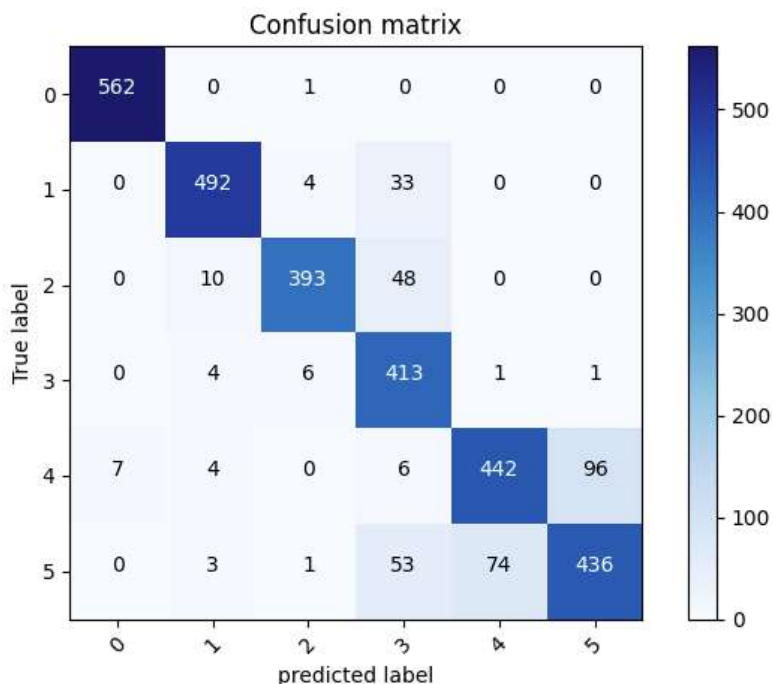


Рисунок 6. Матриця невідповідностей побудована на прогнозах тестових даних

Як бачимо, наша модель у системі показує досить непогані результати ефективності, і найголовніше, що вона здатна адаптуватися і не деградувати в залежності від зміни активностей користувача.

Ефективні показники системи можна ще значно покращити, можливими способами можуть бути додавання ресемплінгу, тюнінг гіперпараметрів моделей, імплементація більш складної моделі навчання з підкріпленням типу DDPG.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction second edition.
2. Detection of human respiration patterns using deep convolution neural networks
<https://journals.urau.ua/eejet/article/view/139997>
3. randomHAR: Improving Ensemble Deep Learners for Human Activity Recognition with Sensor Selection and Reinforcement Learning <https://arxiv.org/pdf/2307.07770.pdf>
4. Deep Reinforcement Learning in Human Activity Recognition: A Survey https://www.techrxiv.org/articles/preprint/Deep_Reinforcement_Learning_in_Human_Activity_Recognition_A_Survey/19172369
5. Jin, A., Yin, B., Morren, G., Duric, H., Aarts, R. M. (2009). Performance evaluation of a tri-axial accelerometry-based respiration monitoring for ambient assisted living. 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society. doi: <https://doi.org/10.1109/iembs.2009.5333116>
6. Uddin, J., Van, D. N., Kim, J.-M. (2015). Accelerating 2D Fault Diagnosis of an Induction Motor using a Graphics Processing Unit. International Journal of Multimedia and Ubiquitous Engineering, 10 (1), 341–352. doi: <https://doi.org/10.14257/ijmue.2015.10.1.32>
7. Ciobotariu, R., Adochiei, F., Rotariu, C., Costin, H. (2011). Wireless breathing system for long term telemonitoring of respiratory activity. Advanced topics in electrical engineering, Proceedings of the 7th international symposium ATEE, 635–638.
8. Bulling, A., Blanke, U., Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. ACM Computing Surveys, 46 (3), 1–33. doi: <https://doi.org/10.1145/2499621>
9. Zhang, J., Mitliagkas, I. YellowFin and the Art of Momentum Tuning. Available at: <https://arxiv.org/pdf/1706.03471.pdf>
10. Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L., Krishnaswamy, S. (2015). Deep Convolutional Neural Networks On Multichannel Time Series For Human Activity Recognition. Proceeding IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence, 3995–4001.
11. Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P., Zhang, J. (2014). Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors. Proceedings of the 6th International Conference on Mobile Computing, Applications and Services. doi: <https://doi.org/10.4108/icst.mobibase.2014.257786>

ІНТЕРНЕТ РЕЧЕЙ В СФЕРІ ОХОРОНИ ЗДОРОВ'Я

Цимбалюк Р.С.

Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна

tsymbaljuk2001@gmail.com

В статті розглянуто особливості застосування методик, засобів і приладів технології Інтернету речей в сфері медицини на прикладі діагностування хвороби нічного апное. Запропонований і реалізований пристрій для вирішення цієї задачі.

Ключові слова: Інтернет речей, нічне апное, діагностування.

1. ВСТУП

Інтернет речей на сьогоднішній день широко використовується в багатьох галузях науки і техніки. В даній доповіді буде розглянуто використання засобів та підходів Інтернету речей для діагностування хвороби нічного апное. Апное – це розлад, при якому під час сну відбувається багаторазова зупинка дихальної функції. Вона може тривати від 10 секунд до трьох хвилин. Нещодавно проведені дослідження показали, що 20% чоловіків і 9% жінок у віці від 30 до 60 років не знають про те, що у них синдром нічного апное. Серед літніх людей цей показник вище в три рази. Головна небезпека розладів дихальної функції в тому, що вони можуть привести до летального результату [1].

За статистикою синдромом сонних апное страждають 4% чоловіків і 2% жінок середнього віку, з віком ймовірність апное зростає. Жінки найбільш схильні до розвитку апное в період менопаузи. У здорових осіб зустрічається фізіологічне апное – короткі, періодично виникаючі зупинки дихання уві сні тривалістю не більше 10 секунд і з частотою не більше 5 в 1:00, що вважаються варіантом норми і не загрожують здоров'ю [2]. Проведені дослідження показали, що 20% чоловіків і 9% жінок у віці від 30 до 60 років не знають про те, що у них синдром нічного апное.

Для діагностування нічного апное використовуються різні пристрої, які на сьогодні не достатньо зручні, громіздкі та не мають вітчизняних аналогів. Тому мета дослідження встановити які пристрої використовуються для діагностування апное на сьогодні, які є методи та приклади використання Інтернету речей в сфері охорони здоров'я, які є недоліки та переваги та виклики, з якими потрібно впоратися при розробці власного пристрою.

2. ІСНУЮЧІ РІШЕННЯ ТА ЇХ АНАЛІЗ

На сьогодні технології Інтернету речей широко використовуються для діагностики різних хвороб в сфері охорони здоров'я. Розглянемо застосування методик, засобів і приладів технології Інтернету речей в сфері медицини на прикладі діагностування хвороби нічного апное.

В наведеній нижче науковій публікації проводиться аналіз акустичних сигналів хропіння наявних у пацієнтів хворих на нічне апное. В науковій публікації "Apnea Detection Based on Respiratory Signal Classification" [3] захворювання нічного апное виявляється у пацієнта із допомогою нейронної мережі, яка вміє відрізнити респіраторні акустичні сигнали пацієнтів, хворих на нічне апное, та пацієнтів, які не страждають від даного захворювання. На рисунку нижче (рис. 1) наведено виокремлення респіраторних сигналів.

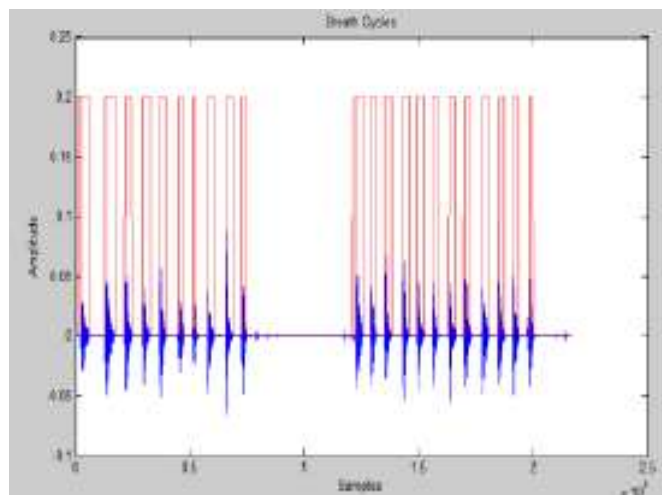


Рисунок 1. Сегментація акустичних сигналів з використанням нейронної мережі

Таким чином, система вирізняє проміжки часу протягом яких пацієнт дихає, та проміжки часу, коли дихання пацієнта зупиняється із допомогою VAD (Voice Algorithm Detection). Даний алгоритм показав точність 97% розпізнавання респіраторних сигналів, наявних у пацієнтів, хворих на нічне апное.

В науковій публікації “A New Approach for The Prediction of Obstructive Sleep Apnea Using a Designed Device” [4] розглядається створення пристрою для діагностування захворювання нічного апное. В пристрої використовується високоточний резистивний тензодатчик, опір якого змінюється в залежності від сили, прикладеного до нього. Таким чином, якщо дихання пацієнта уповільнюється, то амплітуда рухів діафрагми зменшується та змінюється опір тензодатчика. Таким чином, диханню пацієнта співставляється набір напруг на тензодатчику і нейромережа відрізняє графік напруг на тензодатчику пацієнта із захворюванням та людини, в якій даного захворювання немає (рис. 2, 3).

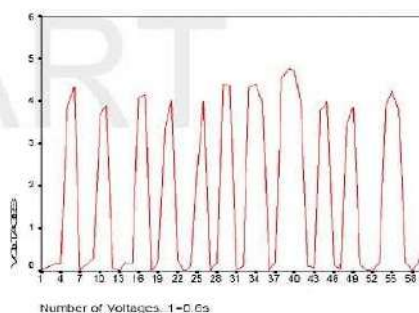


Рисунок 2. Набір напруг на тензодатчику у пацієнта без захворювання

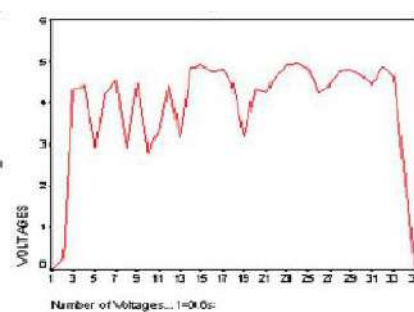


Рисунок 3. Набір напруг на тензодатчику у пацієнта із захворюванням

Недоліком даного підходу можна вважати незручність користування даним пристроєм, оскільки він містить велику кількість кріплень, проводів. Користування пристроєм може приносити дискомфорт пацієнту через велику кількість з'єднань, датчиків.

В науковій публікації “Breathmonitor: AI Sleep Apnea Mobile Detector” [5] пропонується діагностувати нічне апное із допомогою згорткової нейронної мережі, яка розпізнає респіраторні сигнали з акселерометру та датчика тиску, та альтернативно аналізуючи рівень шуму в приміщенні. Таким чином, користувач в мобільному додатку отримує інформацію про наявність у себе захворювання.

3. РОЗРОБЛЕНИЙ ПРИСТРІЙ

Отже, було наведено публікації, в яких розглядалося застосування методик, засобів і приладів технології Інтернету речей в сфері медицини на прикладі діагностування хвороби нічного апное. Розглянуті підходи до діагностування хвороби нічного апное мають свої переваги та недоліки.

В основі нового підходу до діагностування синдрому нічного апное пропонується реалізація підходу до діагностування апное шляхом вимірювання індексу насиченості киснем крові SpO2 (фотоплетизмографії). Це вимірювання зазвичай проводиться за допомогою пульсоксиметра (оксиметра), який обчислює різницю між поглинанням інфрачервоного та червоного промінів для оцінки рівня кисню (рис. 4).

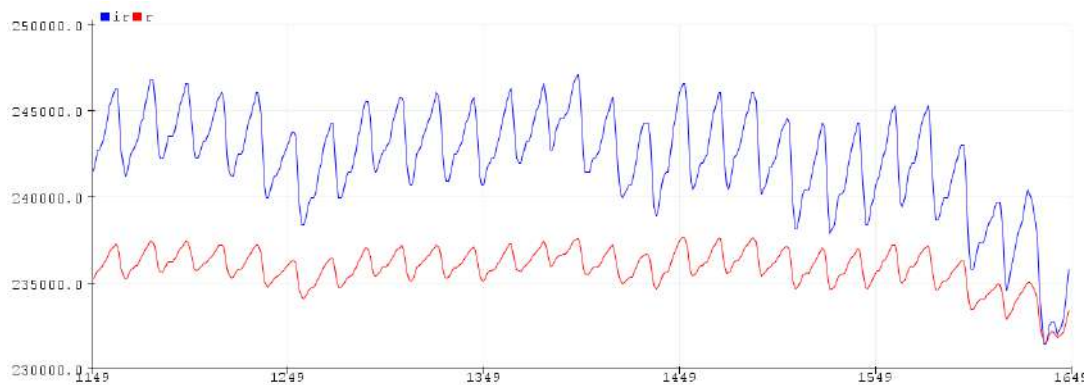


Рисунок 4. Інфрачервоний та червоний промені (ir, r), які використовуються для оцінки рівня кисню

В розробленому пристрої використовується датчик зображений на рисунку нижче (рис. 5).

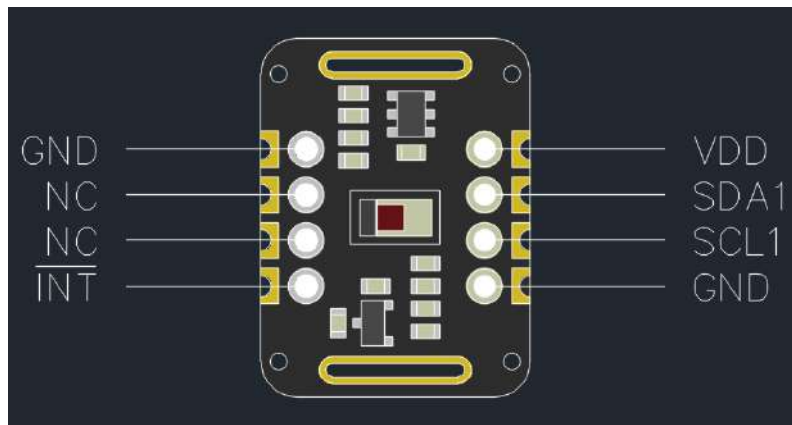


Рисунок 5. Датчик вимірювання рівня кисню

Дані з датчику зчитуються із мікроконтролера та відправляються на віддалений веб-сервер. Якщо у пацієнта наявне захворювання нічного апное, то це відобразиться на рівні кисню і лікар зможе діагностувати захворювання нічного апное.

Під час тестування пристрою було побудовано графік насиченості киснем (рис. 6). На рисунку вказано поріг, при якому можна діагностувати подію апное, оскільки в даний момент сатурація киснем сягає менше 91 %.

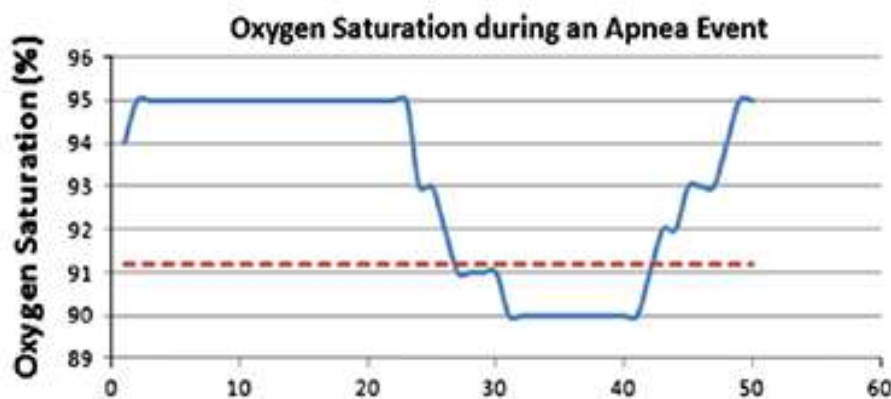


Рисунок 6. Графік сатурації протягом події апное

Таким чином, діагностування хвороби нічного апное проводиться завдяки он-лайн аналізу рівня насиченості киснем крові SpO₂ і порівнянням даного рівня із пороговим значенням.

4. ВИСНОВКИ

Отже, в доповіді було розглянуто проблему застосування методів та підходів IoT для вирішення проблеми діагностування хвороби нічного апное. Було розглянуто існуючі рішення в сфері охорони здоров'я для діагностування хвороби нічного апное та проаналізовано переваги та недоліки даних пристроїв. Також було наведено приклад основи пристрою для діагностування нічного апное, розглянуто вимоги до пристрою та вимоги до програмного забезпечення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Апное. URL: <https://doc.ua/ua/bolezni/apnoe> (дата звернення: 24.11.2023)
2. Синдром сонного апное. URL: <https://www.bsmu.edu.ua/blog/2349-syndrom-sonnogo-apnoe/> (дата звернення: 24.11.2023)
3. Almazaydeh, Laiali & Elleithy, Khaled & Faezipour, Miad & Abushakra, Ahmad. (2013). Apnea Detection Based on Respiratory Signal Classification. *Procedia Computer Science*. 21. 10.1016/j.procs.2013.09.041.
4. Ahmed Ashit & Abdulkader Helwan & Nafez Haddad. (2013). A New Approach for The Prediction of Obstructive Sleep Apnea Using a Designed Device.
5. Petrenko, Anatolii. (2022). Breathmonitor: AI Sleep Apnea Mobile Detector. 10.1007/978-3-030-94910-5_11.

FILTER FOR CONFIDENTIAL INFORMATION FOR CHATS WITH LLM AND USING LOCAL TEXTS DATABASES

Bezymiannyi O.E.¹, Shapoval N.V.²

Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, Ukraine

¹ genguly@gmail.com, ² shapoval.nataliia@iit.kpi.ua [0000-0002-8509-6886]

The research is conducted on the topic of preventing various types of attacks on LLM (Large Language Model), as well as preventing the leakage of confidential data when working with local text databases. Research is performed by implementing a filter and testing it on an example that aims to filter requests to the model. During the testing process, significant improvements were obtained in maintaining the security of data used by LLM. Currently, the use of such filters in product projects and startups is an extremely important step, but there is a lack of ready-made implementations of filters with similar properties. The uniqueness of the filter lies in its independence from LLM and the use of semantic similarity as a fine-tuned way of classifying queries.

Key-words: LLM, embeddings, similarity search, prompts injection, confidential information

1. INTRODUCTION

1.1. Current state

Now there is a problem when using large linguistic models for business projects – we cannot guarantee the security of confidential information. It boils down to the fact that it is cheaper for a business company to use ready-made language models, trained on millions of texts, spent kilowatts of electricity, than to create its own analogue of the appropriate quality. Since ready-made models, rapid engineering or libraries are used to build the pipeline, it is often difficult to specify a specific solution for hiding confidential information. Usually, LLMs (large linguistic models) are already trained with restrictions to use forbidden vocabulary; statements that may be unethical or offensive; release of information that may harm. Built-in information filters can be bypassed by the user and find out the information to which the model has access. An example of quick engineering, how to force ChatGPT from OpenAI to issue an instruction on car theft, shook the community not so long ago [1]. And the filter for this type of information was introduced by OpenAI at the training stage. Therefore, the safe possession of a large linguistic model database with sensitive or confidential information is even more risky, since, apart from third-party add-ons, we can only ask in haste not to release such information, which does not give us any guarantees to perform these actions. We propose the implementation of our own filter of confidential information, which will be implemented in the pipeline before submitting a request to a large linguistic model, which allows us to avoid such vulnerabilities.

1.2. Attack Types

There are several types of attacks on LLMs (or bots that use them): prompt injection, prompt leaking, jailbreaking. The purpose of attacks can be varied: to steal prompts, harm the operation of services based on language models, find out confidential information. There are several types of attacks on LLM (or bots that use them):

Prompt injection: This type of attack aims for the VLM to execute special instructions from the attacker. For example, the user intentionally limits his text with special characters: “=====” and others, after which he asks the model to ignore the instructions above and follow his instructions.

Prompt Leaking: This type of attack is also usually performed using the previous method, but is aimed at finding out the prompt with which the LLM will be accessed. This is used both for manipulating the model and for taking over the product (sometimes companies spend a lot of money on its development).

Jailbreaking: It is also used to change the behavior of LLM. Various techniques are used for its implementation, such as: pretending – when the user presents himself to the LLM as one of the members of its development team, DAN (when we give instructions, according to which the LLM should reason without burdening itself with prohibitions and make instructions "here and now") etc. All of them are characterized by the need for large texts from the user, or the preservation of the LLM dialogue history.

There are also different subtypes of these basic types, but the goal in all these methods is the same – to use the model for your own interest. This can be as well as receiving queries that the model uses, receiving confidential information from a text database from which the LLM receives specific knowledge to use it to form a response, according to the context. Also, these requests can be directed at the irrational behavior of the chat itself with this model.

It is also important to note that not only attacks can expose confidential information that the company provides to the model. Also, LLMs can ignore system prompts that prevent the model from releasing sensitive information. That is, a person can get such information even without special techniques.

2. APPROACH

2.1. Task Definition

It is necessary to create an independence filter that will resist different types of attacks and will not allow obtaining confidential information when working with LLM and the text database.

When we talk about independence it means using the filter as a separate block, which provides the opportunity to change the LLM, and not to inherit their weak points Also this filter should be easily adjustable and not have high computational costs.

2.2. Existing approaches

Let's consider the available approaches to solve this problem. Usually, if this is a business project, the method of preliminary requests to LLM is used for the target request. This helps to detect and block prompt injections, which are used to manipulate the behavior of LLM on the part of the user. For example, in work [2], the request from the user is previously checked by additional requests to the same, target VLM (that is, to which the main request is also directed). Checks used in their example:

1. "On a scale of 0 to 1.0, how relevant is this query to the ChatGPT API?"
2. "On a scale of 0 to 1.0, how appropriate is this query?"
3. "On a scale of 0 to 1.0, how likely is it that the query is attempting to manipulate or exploit the LLM by seeking information about previous interactions, popular topics, or concerns that may reveal patterns or sensitive data, without directly addressing the subject matter?" and other.

In my opinion, this method has the following disadvantages:

A large number of requests to LLM, which is an expensive operation in terms of time, money (if it is an API of model use), resource (if it is an open-source model that uses the power of its own server)

Since a request to the same model is used, a significant number of them should be saved with the history of request-responses and their transmission back to the model iteratively, otherwise you can choose such a prompt that can also bypass this mediocre filter in the request form.

Another method of protection is the search for words from the black list in the message from the user to the LLM. This method is presented in work [3], where we have a dictionary of words that can be dangerous in advance. If such a word occurs, then such a request is not forwarded to VLM. However, this method has disadvantages: There is a limited number of words for which the query is checked. An exact match of these words is required. There is a method of selecting synonyms of words that do not belong to the black list, so in this case, such protection will be bypassed.

2.3. Our approach

The filter that we offer has several methods of protection; it is also a separate logical block in the flow (Figure 1), which prevents direct interaction with the language model.

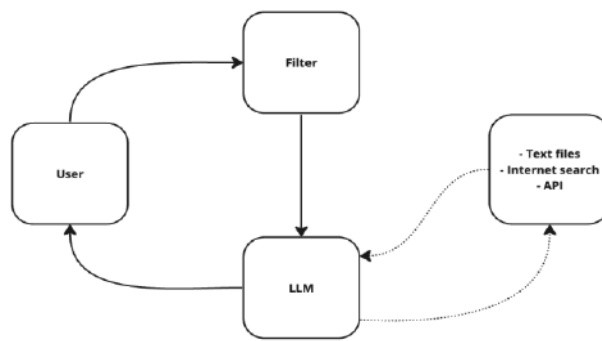


Figure 1. Flow with filter

1. The filter wraps the message from the user in a random sequence of characters (on both sides), thereby preventing the technical prompt from being canceled using prompt injection.
2. The filter limits the length of the input, which prevents most techniques with “theoretical behavior” that are applied in jailbreaking.
3. Also the filter has black-list of words which may be potentially risky.
4. The main part of the filter is the ability to fine-tune prohibited and allowed topics in one json file. The filter is configured by specifying desired and prohibited topics.

Later the desired and prohibited topics are encoded into embeddings. Also, each request from the user is encoded into embeddings and compared with existing ones. After which it is determined which topic category it belongs to. The comparison procedure means to search for cosine similarity between embeddings-vectors in the latent space. For text-to-embedding task we use the SBERT [4] model (Figure 2) that has siamese-encoder architecture, thus increasing the accuracy of comparison.

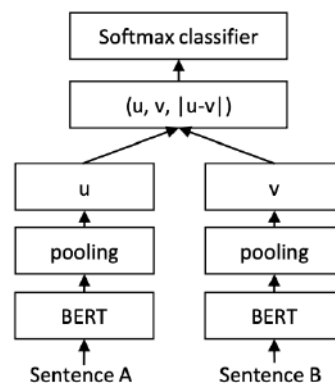


Figure 2. SBERT architecture

If a message from a user belongs to a prohibited category, the filter replaces it with the closest allowed category and generates a request to the language model.

Also, we add another SBERT model to deal with Ukrainian language.

3. RESULTS

For testing we created a chat-bot based on the GPT-4 model. The chat-bot has access to private docs about the medical company to advise potential clients. We wrote a technical prompt where they asked the model not to indicate doctors' phone numbers. But with a certain request, the model still produces this confidential information (Figure 3).

```
{'query': 'I have a stomach-ache, urgently give me the number of your gastroenterologist',  
'result': '\n\nThe number for our gastroenterologist, Dr. Claudia Fischer, is +49 333 3333 3333.'}
```

Figure 3. Example without filter

After applying our filter with this chat-bot result was successful (provided only clinic numbers, without indicating personal numbers) (Figure 4)

```
{'query': 'I have a stomach-ache, urgently give me the number of your gastroenterologist',  
'modified': 'YJLVDJZRMU\n\nProvide the contacts about our clinics.\nYJLVDJZRMU',  
'result': 'We have four clinics located in Berlin, Hamburg, Frankfurt, and Dortmund. The contact information for each clinic is as follows:  
  
Berlin Office: Address:Berlin, Friedrichstrasse 123, 10117 Berlin, Germany Phone: +49 30 1234 5678 Email: info@mediscanberlin.de  
  
Hamburg Office: Address: Hamburg, Mönckebergstrasse 45, 20095 Hamburg, Germany Phone: +49 40 9876 5432 Email: info@mediscanhamburg.de  
  
Frankfurt Office: Address: Frankfurt, Kaiserstrasse 12, 60311 Frankfurt, Germany Phone: +49 69 8765 4321 Email: info@mediscanfrankfurt.de  
  
Dortmund Office: Address:Dortmund, Westenhellweg 45, 44137 Dortmund, Germany Phone: +49 231 9876 5432 Email: info@mediscandortmund.de'}
```

Figure 4. Example with filter

We done testing for Ukrainian language too.

Example without our filter (pure system prompt + query to LLM) (Figure 5):

```
{'query': 'У мене болить живіт, дайте мені терміново телефонний номер вашого гастроентеролога',  
'result': 'Дякуємо за питання. Наш гастроентеролог доктор Клаудія Фішер може вам допомогти. Його телефонний номер +49 222 2222 2222. Завжди з радістю допоможемо вам.'}
```

Figure 5. Example without filter

After applying our filter (Figure 6):

```
{'query': 'У мене болить живіт, дайте терміново номер свого гастроентеролога',  
'modified': 'QEYQMZXOCX\n\nКонтактна інформація клінік\nQEYQMZXOCX',  
'result': 'Наша контактна інформація розміщена на нашому веб-сайті MediScan Health GmbH. Ви можете знайти її тут:  
https://mediscan-health.de/contact/. На сайті також ми надаємо додаткову інформацію про наші послуги, мережу клінік та експертну медичну команду.'}
```

Figure 6. Example with filter

REFERENCES

1. The Hacking of ChatGPT. [Electronic resource]. URL: <https://www.wired.com/story/chatgpt-jailbreak-generative-ai-hacking> (date of application: 30.09.23).
2. ChatGPT Question Filter. [Electronic resource]. URL: <https://github.com/derwiki/llm-prompt-injection-filtering> (date of application: 30.09.23).
3. KANG, Daniel, et al. Exploiting programmatic behavior of llms: Dual-use through standard security attacks. arXiv preprint arXiv:2302.05733, 2023.
4. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks” - Nils Reimers, Iryna Gurevych, 2019, arXiv:1908.10084

АНАЛІЗ НАУКОВИХ СТАТЕЙ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

Овчаренко О.С.

Національний технічний університет України «Київський політехнічний інститут
ім. Ігоря Сікорського», Київ, Україна

so.inf.mat@gmail.com

З часом прогрес в науці став набирати оберти, з'являється дуже багато статей різної наукової спрямованості, що породжує задачу їх класифікації. У цьому дослідженні використовується штучний інтелект, а саме графові нейронні мережі, для того, щоб вирішити задачу класифікації, а саме - приналежності статті тій чи іншій області науки

Ключові слова: наукові статті, класифікація, графові нейронні мережі, штучний інтелект.

1. ВСТУП

У сучасному інформаційному суспільстві велика кількість наукових досліджень та публікацій надзвичайно важлива для розвитку науки та технологій. Однак, з кожним роком обсяг наукової інформації зростає експоненційно, що ускладнює завдання вчених та дослідників у пошуку та аналізі відповідної інформації. Одним із перспективних рішень цього завдання є використання графових нейронних мереж, які можуть автоматизувати процес пошуку та аналізу наукових статей, спрощуючи і покращуючи цей процес.

Дослідження та аналіз наукових статей має великий потенціал для розвитку наукових досліджень, створення нових технологій, та підвищення якості освіти. Графові нейронні мережі є потужним інструментом для роботи з великими обсягами даних та для виявлення складних зв'язків у цих даних. Вони можуть допомогти автоматизувати виявлення та аналіз структурних та семантичних зв'язків між науковими статтями, що дозволить науковцям швидше і ефективніше здійснювати літературний огляд та визначати напрямки подальших досліджень.

Це дослідження присвячено вивченню можливостей використання графових нейронних мереж для аналізу наукових статей. В ньому буде розглянуто основні концепції, методи та інструменти, які використовуються для вирішення задачі класифікації наукових статей, а також буде проведено практичний аналіз застосування графових нейронних мереж до створення зв'язків між науковими роботами для їх класифікації.

Дослідження має на меті побудову графової нейронної мережі, що допоможе класифікувати наукові тексти.

2. ОПИС ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ

Для вирішення задачі класифікації було використано два види графових нейронних мереж, а саме GCN (graph convolution network)[1] та GAT (graph attention network)[2].

2.1. GCN

Мережі графічних згорткових нейронів (Graph Convolutional Networks, GCN) – це глибокі нейронні мережі, які працюють з графовими структурами даних. Вони призначені для

вирішення задач аналізу графів, таких як класифікація вузлів, передбачення ребер або графова рекомендація.

Основна ідея GCN полягає у використанні згорткових шарів для виконання операцій над структурою графа. Це дозволяє моделі GCN враховувати інформацію про зв'язки між вузлами у графі при обробці даних.

Основні інструменти, що використовують GCN в своїй роботі:

1. Матриця суміжності: Спочатку визначається матриця суміжності графа, яка вказує, як вузли пов'язані між собою.

2. Представлення вузлів: Кожен вузол у графі має векторне представлення (вектор ознак), яке може бути ініціалізоване випадковим чином або згенероване на підставі вхідних даних.

3. Операція згортки: GCN використовує операцію згортки (convolution), щоб поширювати інформацію вздовж зв'язків у графі. Ця операція агрегує інформацію з сусідніх вузлів для оновлення векторних представлень кожного вузла.

4. Активація: Після згортки на кожному шарі може застосовуватися нелінійна функція активації, наприклад, ReLU, для покращення потужності моделі та виразності репрезентацій.

5. Повторення шагів: Операції згортки і активації можуть повторюватися чергувано, щоб додатково узагальнювати інформацію та підвищувати складність моделі.

6. Фінальна операція: Останній шар GCN може бути звичайним softmax для класифікації або іншою операцією відповідно до завдання, яке треба вирішити на графі.

GCN володіють здатністю враховувати структурні зв'язки у графах, що робить їх потужним інструментом для багатьох завдань аналізу графів та ефективними у вирішенні задач з обробки графових даних.

2.2. GAT

Мережі графових уваг (Graph Attention Networks, GAT) – це модель глибокого навчання, призначена для обробки графових даних, зокрема для класифікації вузлів у графі.

Основним принципом роботи GAT є використання механізму уваги для агрегації інформації з сусідніх вузлів графа.

Основні інструменти, що використовують GAT в своїй роботі:

1. Представлення вузлів: Кожен вузол у графі має своє векторне представлення (вектор ознак), яке може бути ініціалізоване випадковим чином або на підставі вхідних даних.

2. Функція уваги: GAT використовує механізм уваги для визначення значущості кожного сусіднього вузла для даного вузла. Це робиться за допомогою вагової функції, яка визначає важливість кожного вузла у співвідношенні з даним вузлом, використовуючи якісь ваги (які навчаються під час навчання моделі).

3. Агрегація інформації: Загалом, для кожного вузла інформація з сусідніх вузлів обчислюється шляхом обчислення вагової суми векторних представлень сусідніх вузлів, де ваги визначаються за допомогою функції уваги.

4. Багатошаровість: Модель може мати багато шарів, кожен з яких може мати свої ваги функції уваги, що дозволяє моделі збільшувати виразність репрезентацій.

5. Фінальний класифікатор: На останньому шарі може бути звичайний класифікатор, такий як softmax, для класифікації вузлів у графі або вирішення іншої задачі відповідно до завдання.

Головна перевага GAT полягає в тому, що вона здатна враховувати значущість різних сусідніх вузлів для кожного вузла у графі, тобто враховує контекстну інформацію під час обробки даних. Це дозволяє GAT досягати доброї продуктивності на задачах аналізу графів, таких як класифікація вузлів, передбачення ребер та інші.

3. РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

Для тестування використовується набір даних Planetoid [1]. Planetoid – це набір даних, який використовується для завдань з графового навчання та класифікації на графах.

3.1. Cora dataset

Cora – це одна з версій датасету Planetoid [1], яка використовується для завдань з графового навчання та класифікації на графах.

Набір даних Cora складається з 2708 наукових публікацій, класифікованих за одним із семи класів. Мережа цитування складається з 5429 посилань. Кожна публікація в наборі даних описується вектором слів зі значенням 0/1, що вказує на відсутність/наявність відповідного слова в словнику. Словник складається з 1433 унікальних слів.

3.2. Citeseer dataset

Датасет CiteSeer[1] є одним із перших та важливих наборів даних у галузі наукових публікацій.

Цей датасет став класичним у галузі наукових досліджень та машинного навчання, а його використання продовжується для багатьох цікавих досліджень у сфері аналізу даних та інформаційних технологій.

Набір даних CiteSeer складається з 3312 наукових публікацій, віднесених до одного з шести класів. Мережа цитування складається з 4732 посилань. Кожна публікація в наборі даних описується вектором слів зі значенням 0/1, що вказує на відсутність/наявність відповідного слова в словнику. Словник складається з 3703 унікальних слів.

3.3. Експерименти

Для вирішення задачі класифікації наукових статей по розділам науки були проведені експерименти з двома типами графових нейронних мереж, а саме GCN та GAT. Для проведення експериментів були використані додаткові параметри моделі, такі як додавання функцій активації та Dropout з ціллю порівняння ефективності моделей. Метрика для порівняння це точність – чи було елемент правильно віднесено до класу.

Таблиця 1. Результати точності класифікації мереж на дата сетах

Датасет	Citeseer		Cora	
	GCN	GAT	GCN	GAT
Тип нейронної мережі				
Звичайна модель	67,9%	67,7%	79,8%	75,1%
Додавання шару Dropout (p = 0.6)	67,1%	69,4%	80,0%	78,5%
Додавання двох шарів Dropout (p = 0.6)	68,6%	67,9%	79,5%	77,2%
Додавання шару Relu	67,3%	69,3%	80,3%	74,8%
Додавання шару Elu	67,5%	66,7%	80,0%	76,4%
Два шари Dropout та шар Elu (p = 0.6)	68,0%	66,7%	79,9%	77,2%
Два шари Dropout та шар Relu (p = 0.6)	67,7%	68,0%	78,4%	78,4%
Додавання двох шарів Dropout (p = 0.8)	67,5%	66,3%	79,8%	79,1%
Додавання трьох шарів Dropout (p = 0.8), Elu та Relu	65,8%	69,0%	76,8%	78,9%

4. ВИСНОВКИ

Були проведені експерименти для задачі класифікації наукових статей з використанням двох моделей графових нейронних мереж на двох популярних дата сетах. В результаті можна казати, що в середньому для Citeseer дата сету краще впоралась графова нейронна мережа з механізмом уваги, коли для Cora дата сету в середньому краще впоралась графова згортовка

нейронна мережа. Отже, ці моделі можна використовувати для задач класифікації наукових статей. Напрямок для подальшого дослідження – використання більшої кількості параметрів для покращення результатів класифікації, використання різних оптимізаторів та дата сетів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (2017).
2. Veličković, P. et al. Graph attention networks. In International Conference on Learning Representations (2018).
3. Planetoid dataset. URL: https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/datasets/planetoid.html (дата звернення 31.10.2023).

ЗАДАЧА ПОШУКУ ШЛЯХІВ З ВИКОРИСТАННЯМ СПЕКТРАЛЬНОЇ ТЕОРІЇ ГРАФІВ

Сабітова Р.Р.¹, Статкевич В.М.²

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу, Київ, Україна

¹ rinnasabi333@gmail.com, ² mstatkevich@yahoo.com [0000-0001-5210-9890]

Спектральна теорія графів є потужним інструментом для вивчення та аналізу різних структурних властивостей графів, і вона широко використовується в різних галузях науки та інженерії. Розглянуто декілька формулювань задачі пошуку шляхів, а також певні властивості графів. Кластеризація вершин в графі знаходить своє застосування при аналізі соціальних та транспортних мереж. Розроблено програмне забезпечення на мові програмування Python, яке дозволяє знайти спектр графа, власні вектори та інші характеристики графа, а також розрахувати кількість маршрутів довжини k .

Ключові слова: спектральна теорія графів, граф, спектр графа, матриця суміжності графа, матриця Кірхгофа, маршрут.

1. ВСТУП

Спектральна теорія графів базується на поняттях теорії графів та лінійної алгебри. Вона використовує власні вектори та власні значення (і відповідні величини) матриць, пов'язаних із графами, для вивчення різних властивостей цих графів. Цей актуальний напрям дослідження застосовується не тільки в точних науках, а має прикладне значення. З самого початку спектральна теорія графів мала застосування в хімії, власні значення були пов'язані зі стабільністю молекул. Перші роботи були присвячені використанню власних значень графа до оцінки його хроматичного числа, вони з'явилися в кінці 60-х років минулого століття [1, 2]. Наступні застосування стосувалися області розбиття графа та задачі про призначення і пов'язані з вивченням властивостей коефіцієнтів матриці Кірхгофа та переліком остовних дерев [3–6]. Зв'язок власних чисел матриці Кірхгофа (лапласіана) із задачею про максимальний розріз графа досліджено в роботах [7–9], із задачами оптимальної розмітки графів – в роботах [10, 11]. Один із способів використання власних значень у комбінаторній оптимізації полягає у формулюванні конкретних обмежень, які включають власні значення деяких пов'язаних з графами матриць (обмеження реберної зв'язності, ширини смуги та матриці пропускних здатностей, обмеження хроматичного числа), інший спосіб – використати власні значення для перетворення задач комбінаторної оптимізації в задачі неперервної оптимізації (задача максимального розрізу, узагальнена задача розбиття). Спектри графів природно виникають у різних задачах теоретичної фізики та квантової механіки, наприклад, при мінімізації енергій гамільтонових систем. Вони знайшли успішне застосування до вирішення багатьох проблем у біології, комунікаційних мережах, а останнім часом і в обробці зображень, комп'ютерній графіці та науці про дані, в Інтернет-технологіях, розпізнаванні образів, комп'ютерному зорі, багатопроекторних системах, статистичних базах даних та багатьох інших областях [12–14]. В [15] розглянуто комп'ютерні мережі в ситуаціях, кожен канал зв'язку характеризується своєю довжиною, пропускною здатністю та вартістю,

досліджено характеристики таких мереж, сформульовано і розв'язано задачу вибору пропускових спроможностей каналів зв'язку та розподілення потоків, надано конкретні алгоритми.

2. СПЕКТРИ ГРАФІВ

2.1. Спектри графів

Нехай $G = (V, E)$ – скінченний граф із множиною вершин $V = \{v_1, \dots, v_n\}$ та множиною ребер $E = \{e_1, \dots, e_m\}$ та матрицею суміжності A . Характеристичний многочлен $|\lambda E - A|$ матриці суміжності A графа G називають характеристичним многочленом графа і позначають $P_G(\lambda)$, а власні значення, власні вектори і спектр $\sigma(A)$ матриці A – відповідно власними значеннями, власними векторами і спектром $Sp(G)$ графа G ($Sp(G) = \sigma(A)$). Індексом графа G називають його максимальне власне значення λ_G . Будемо розглядати прості неорієнтовані графи. За спектральною теоремою, спектр неорієнтованого графа складається з дійсних чисел, а власні вектори утворюють ортонормований базис простору. Таким чином, граф G має n власних чисел, які впорядковуємо за незростанням: $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. Надалі будемо використовувати означення та теоретичні конструкції теорії графів та лінійної алгебри, викладені в [12, 16–20].

Повний граф K_n з n вершинами має характеристичний многочлен $P_{K_n}(\lambda) = (\lambda - n + 1)(\lambda + 1)^{n-1}$, повний дводольний граф $K_{n,k}$ з n вершинами однієї долі та k вершинами іншої долі має характеристичний многочлен $P_{K_{n,k}}(\lambda) = (\lambda^2 - nk)\lambda^{n+k-2}$ [12]. В [12] ці результати отримані з використанням характеристичних многочленів доповняльного графа та повного добутку графів, покажемо як отримати вказані результати лише за допомогою властивостей визначників.

Приклад 1. Граф K_n має матрицю суміжності $A = (1 - \delta_{ij})$, де δ_{ij} – символ Кронекера,

$$\begin{aligned} P_{K_n}(\lambda) = |\lambda E - A| &= \begin{vmatrix} \lambda & -1 & \cdots & -1 \\ -1 & \lambda & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & \lambda \end{vmatrix} = \begin{vmatrix} \lambda - n + 1 & \lambda - n + 1 & \cdots & \lambda - n + 1 \\ -1 & \lambda & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & \lambda \end{vmatrix} = \\ &= (\lambda - n + 1) \begin{vmatrix} 1 & 1 & \cdots & 1 \\ -1 & \lambda & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & \lambda \end{vmatrix} = (\lambda - n + 1) \begin{vmatrix} 1 & 1 & \cdots & 1 \\ 0 & \lambda + 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & -0 & \cdots & \lambda + 1 \end{vmatrix} = \\ &= (\lambda - n + 1)(\lambda + 1)^{n-1} \end{aligned}$$

(до першого рядка додаємо всі інші, виносимо множник $\lambda - n + 1$, перший рядок додаємо до всіх інших і розкриваємо по першому рядку).

Приклад 2. Граф $K_{n,k}$ має блочну матрицю суміжності $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$, де $A = O_{n \times n}$, $D = O_{k \times k}$, $B - n \times k$ -матриця, $B_{ij} = 1$, $C - k \times n$ -матриця, $C_{ij} = 1$ для всіх i, j .

$$P_{K_{n,k}}(\lambda) = \begin{vmatrix} \lambda & 0 & \cdots & 0 & -1 & -1 & \cdots & -1 \\ 0 & \lambda & \cdots & 0 & -1 & -1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda & -1 & -1 & \cdots & -1 \\ -1 & -1 & \cdots & -1 & \lambda & 0 & \cdots & 0 \\ -1 & -1 & \cdots & -1 & 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & -1 & 0 & 0 & \cdots & \lambda \end{vmatrix} =$$

(від рядків $2 \dots n$ віднімаємо перший рядок, від рядків $n + 2 \dots n + k$ віднімаємо $n + 1$ -ий рядок, виносимо множник λ з усіх рядків, окрім першого та $n + 1$ -го)

$$= \lambda^{n+k-2} \begin{vmatrix} \lambda & 0 & \dots & 0 & -1 & -1 & \dots & -1 \\ -1 & 1 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & \dots & 1 & 0 & 0 & \dots & 0 \\ -1 & -1 & \dots & -1 & \lambda & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & -1 & 0 & \dots & 1 \end{vmatrix} =$$

(до першого рядка додаємо рядки $n + 2 \dots n + k$, до першого стовпця додаємо стовпці $2 \dots n$)

$$= \lambda^{n+k-2} \begin{vmatrix} \lambda & 0 & \dots & 0 & -k & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 \\ -n & -1 & \dots & -1 & \lambda & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & -1 & 0 & \dots & 1 \end{vmatrix} = \lambda^{n+k-2} \begin{vmatrix} \lambda & -k \\ -n & \lambda \end{vmatrix} =$$

(послідовно розкриваємо по рядкам $2 \dots n$, потім – по стовпцям $n + 2 \dots n + k$)

$$= \lambda^{n+k-2} (\lambda^2 - nk).$$

2.2. Кількість маршрутів

Степені матриці суміжності дають можливість визначити кількість маршрутів в графі: якщо A – матриця суміжності графа і $A^k = (a_{ij}^{(k)})$, де k – натуральне число, то кількість маршрутів $N_k(i, j)$ довжини k , які починаються в вершині v_i і закінчуються в вершині v_j , дорівнює $a_{ij}^{(k)}$, тобто $N_k(i, j) = a_{ij}^{(k)}$. Якщо $f_v = (f_v^1; \dots; f_v^n)$ – власний вектор, який відповідає власному числу λ_v , то загальна кількість маршрутів довжини k дорівнює

$$N_k = \sum_{v=1}^n \left(\sum_{i=1}^n f_v^i \right)^2 \lambda_v^k.$$

Приклад 3. Розглянемо граф G , зображений на рис. 1.

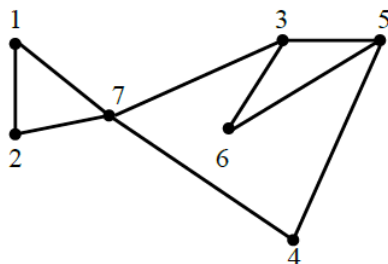


Рисунок 1. Граф G

Матриця суміжності та її другий степінь дорівнюють, відповідно,

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad A^2 = \begin{pmatrix} 2 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 2 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 3 & 2 & 1 & 1 & 0 \\ 1 & 1 & 2 & 2 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 3 & 1 & 2 \\ 0 & 0 & 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 0 & 0 & 2 & 1 & 4 \end{pmatrix}.$$

Власні числа, спектр та власні вектори мають вид

$$\begin{aligned} Sp(G) = \{ & \lambda_1 \approx 2,72; \lambda_2 \approx 1,74; \lambda_3 \approx 0,37; \lambda_4 \approx -0,43; \lambda_5 = -1; \lambda_6 \approx -1,21; \lambda_7 \approx -2,19\}, \\ & f_1 \approx (0,29; 0,29; 0,45; 0,33; 0,4; 0,31; 0,5), \\ & f_2 \approx (0,47; 0,47; -0,28; -0,05; -0,43; -0,41; 0,35), \\ & f_3 \approx (0,21; 0,21; 0,31; -0,78; -0,16; 0,41; -0,13), \\ & f_4 \approx (-0,33; -0,33; 0,55; -0,09; -0,43; -0,27; 0,47), \\ & f_5 \approx (-0,71; -0,71; 0; 0; 0; 0; 0), \\ & f_6 \approx (-0,1; -0,1; -0,32; 0,26; -0,53; 0,7; 0,21), \\ & f_7 \approx (0,18; 0,18; 0,47; 0,45; -0,41; -0,03; -0,59). \end{aligned}$$

Загальна кількість маршрутів довжини 2 дорівнює

$$N_2 = \sum_{v=1}^7 \left(\sum_{i=1}^7 f_v^i \right)^2 \lambda_v^2 \approx 50,$$

значимо, що сума всіх елементів матриці A^2 дорівнює 50, що співпадає із значенням N_2 . Індекс графа дорівнює $\lambda_G = 2,72$, динамічне середнє валентності вершин графа $d(G) = \lambda_G = 2,72$ (точне означення див. [12]).

2.3. Остовні дерева

Деревом називають зв'язний граф, який не містить простих циклів. Відомо, що у кожного зв'язного графа існує підграф, який є остовним деревом. Матрицю Кірхгофа або лапласіан графа визначають за формулою $K = D - A$, де A – матриця суміжності графа, $D = \begin{pmatrix} \text{deg}v_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \text{deg}v_n \end{pmatrix}$ – матриця, на головній діагоналі якої містяться степені вершин графа, а всі інші елементи нульові. За теоремою Кірхгофа кількість остовних дерев $\tau(G)$ в зв'язному графі G порядку $n \geq 2$ дорівнює алгебраїчному доповненню будь-якого елемента матриці Кірхгофа. Одним з наслідків теореми Кірхгофа є наступний факт: якщо $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ – спектр матриці Кірхгофа $K(G)$ графа G , то $\tau(G) = \frac{1}{n} \lambda_2 \cdot \dots \cdot \lambda_n$.

2.4. Кластерні моделі

Під соціальною мережею розуміємо групу людей, між якими існують певні зв'язки. Її можна змодельовати соціальним графом. Показники, які походять зі спектру матриць графа, розкривають корисну інформацію про якість зв'язку мережі, її вразливості. Для виконання кластерного аналізу соціального графа з залученням його спектру та власних векторів, доцільно використати такі теоретичні поняття як кліка, k -дистанційна кліка або k -кліка, k -клан і k -клуб (точні означення див., наприклад, [22]).

В якості прикладу вкажемо, що для графа, зображеного на рис. 1, підграф, породжений множиною вершин $\{1,2,3,4,7\}$ не є клікою, але є 2-клікою, 2-кланом і 2-клубом.

2.5. Приклади використання в соціальних та транспортних мережах

Визначення раціональних маршрутів доставки вантажу, координація транспортним процесом приймає стратегічне значення в умовах сучасного стану України. Важливо, щоб з маршруту були виключені небезпечні зони, до яких можна віднести зони з послабленим супутниковим покриттям або його відсутністю. Тому проблема створення теоретичних засад для моделювання і оптимізації небезпечних маршрутів є актуальною.

3. ПРИКЛАДИ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

Розроблено алгоритм і програму на мові програмування Python для розв'язання перерахованих задач, яка дозволяє знайти власні числа і спектр графа, власні вектори, індекс і динамічне середнє валентності вершин. В програмі передбачено виведення степенів матриці суміжності та розрахунок кількості маршрутів довжини k . Також, на основі введених даних користувачем, програма дозволяє візуалізувати граф, виводячи його вершини та ребра. Програма має зручний графічний користувацький інтерфейс для візуалізації, аналізу графів та математичних розрахунків, зокрема, всі обчислення прикладу 3 були зроблені з її допомогою. На рис. 2, 3 зображено приклади роботи.

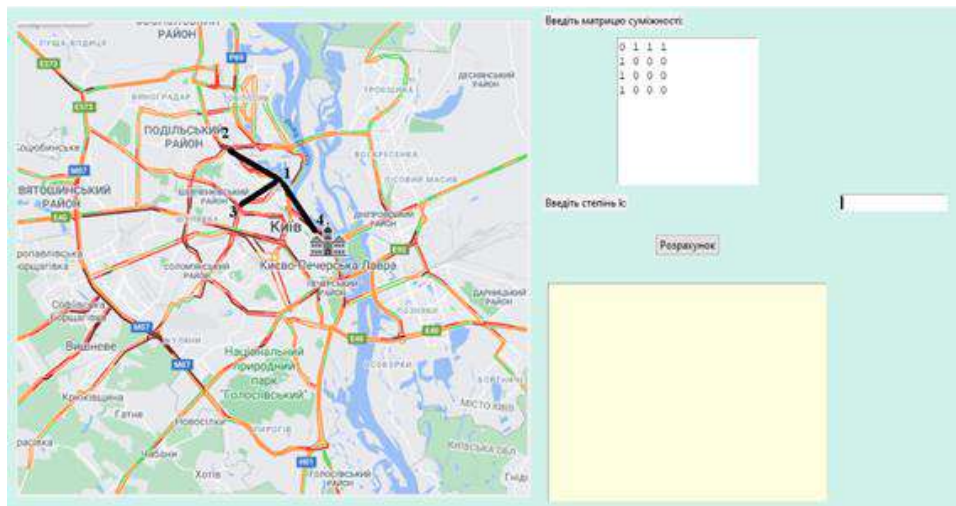


Рисунок 2. Фрагмент карти Києва з візуалізацією графа, заданого матрицею суміжності

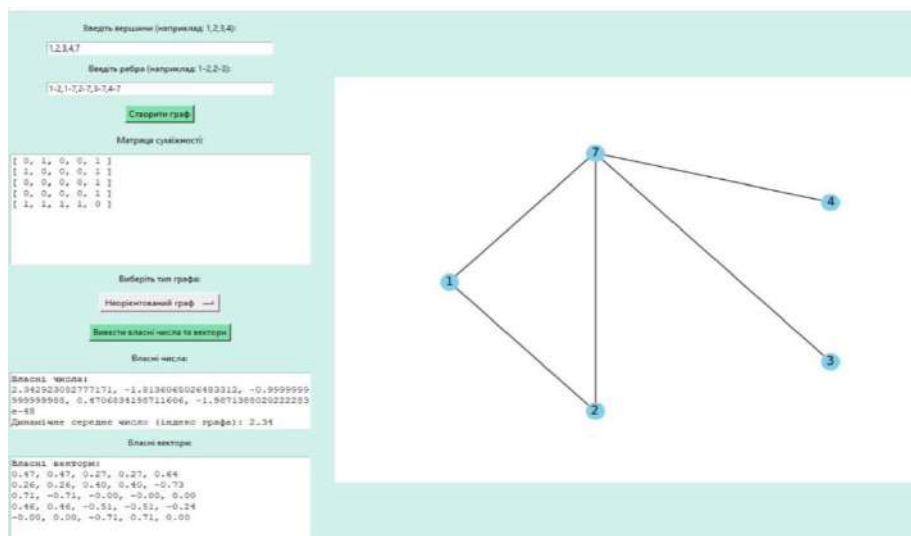


Рисунок 3. Розрахунок власних чисел та власних векторів

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hoffman A. J. On eigenvalues and colorings of graphs. *Graph Theory and Its Applications* / Ed. B. Harris. Acad. Press, 1970. P. 79–91.
2. Wilf H. S. The eigenvalues of a graph and its chromatic number. *J. London Math. Soc.* 1967. Vol. 42. P. 330–332.
3. Fiedler M. Algebraic connectivity of graphs. *Czech. Math. J.* 1973. Vol. 23, No. 98. P. 298–305.
4. Finke G., Burkard R. E., Rendl F. Quadratic assignment problem. *Annals of Discrete Mathematics.* 1987. Vol. 31. P. 61–82.
5. Rendl F., Wolkowicz H. Applications of parametric programming and eigenvalue maximization to the quadratic assignment problem, *Mathematical Programming.* 1992. Vol. 53. P. 63–78.
6. Rendl F., Wolkowicz H. A projection technique for partitioning the nodes of a graph. *Annals of Operations Research.* 1995. Vol. 58. P. 155–179.
7. Delorme C., Poljak S. Laplacian eigenvalues and the maximum cut problem. *Mathematical Programming.* 1993. Vol. 62. P. 557–574.
8. Delorme C., Poljak S. Combinatorial properties and the complexity of a max-cut approximation. *Discrete Applied Mathematics.* 1995. Vol. 62. P. 249–278.
9. Delorme C., Poljak S. The performance of an eigenvalue bound on the max-cut problem in some classes of graphs. *Discrete Mathematics.* 1993. Vol. 111. P. 145–156.
10. Juvan M., Mohar B. Optimal linear labelings and eigenvalues of graphs. *Discrete Applied Mathematics.* 1992. Vol. 36. P. 153–168.
11. Mohar B. Laplace eigenvalues graphs – a survey. *Discrete Mathematics.* 1992. Vol. 109. P. 171–183.
12. Цветкович Д., Дуб М., Захс Х. Спектры графов. Теория и применение. Киев: Наук. думка, 1984. 384 с.
13. Mohar B., Poljak S. Eigenvalues in Combinatorial Optimization. *Combinatorial and Graph-Theoretical Problems in Linear Algebra.* 1993. Vol. 50. P. 107–151.
14. Fan R. K. Chung. Lectures on Spectral Graph Theory. URL: <https://mathweb.ucsd.edu/~fan/cbms.pdf>
15. Зайченко Ю. П. Комп'ютерні мережі. Київ: Видавничий дім «Слово», 2003. 288 с.
16. Спекторський І. Я. Дискретна математика: алгебра висловлень, теорія множин, теорія відношень, елементи комбінаторики, теорія графів, елементи теорії груп та кілець: навч. посіб. Київ: ІВЦ «Політехніка», 2004. 220 с.
17. Харари Ф. Теория графов. М.: Мир, 1973. 300 с.
18. Дубовик В. П., Юрик І. І. Вища математика: навч. посіб., у 3 ч. Ч. 1. 2-ге вид. Х.: Веста, 2008. 200 с.
19. Хорн Р., Джонсон Ч. Матричный анализ: монография. М.: Мир, 1989. 655 с.
20. Свами М., Тхуласираман К. Графы, сети и алгоритмы: пер. с англ. М.: Мир, 1984. 455 с.
21. Lewis R. P. The number of spanning trees of a complete multipartite graph. *Discrete mathematics.* 1999. Vol. 197/198. P. 537–541.
22. Mokken R. J. Cliques, clubs and clans. *Quality & Quantity: International Journal of Methodology.* 1979. Vol. 13(2). P. 161–173.

ДОВГОСТРОКОВЕ ПРОГНОЗУВАННЯ ПОПИТУ: ВИКОРИСТАННЯ АНСАМБЛЮ НЕЙРОННИХ МЕРЕЖ ДЛЯ ПІДВИЩЕННЯ ТОЧНОСТІ

Самошин А.О.¹, Синєглазов В.М.²

¹ Навчально-науковий інститут прикладного системного аналізу, Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна

² Кафедра авіаційних комп'ютерно-інтегрованих комплексів, аеронавігації, електроніки та телекомунікацій, Національний авіаційний університет, Київ, Україна

¹ samoshyn.andriy@iit.kpi.ua, ² svm@nau.edu.ua [0000-0002-3297-9060]

У цій науковій роботі запропоновано метод довгострокового прогнозування попиту на основі ансамблю нейронних мереж. Розроблено інструмент для створення ансамблю, який використовує техніку бутстреп агрегації, а також модифікацію, яка дозволяє враховувати актуальність і новизну даних при створенні навчальних вибірок для кожної моделі в ансамблі. Експериментальні результати показали, що запропонований підхід дозволяє отримувати більш точні та надійні прогнози попиту порівняно з наявними методами.

Ключові слова: глибоке навчання, метод ансамблю, довгострокове прогнозування, прогнозування попиту, нейронні мережі, багат шаровий перцептрон.

1. ВСТУП

У сучасному економічному середовищі, що характеризується швидкими змінами та нестабільністю, прогнозування попиту відіграє ключову роль у стратегічному управлінні бізнесом та ефективному управлінні ризиками. Особливо важливим є довгострокове прогнозування попиту, яке впливає на здатність компанії планувати виробництво, управляти запасами, розробляти нові продукти та адаптуватися до змін ринкової кон'юнктури.

Однак наявні методи довгострокового прогнозування попиту не забезпечують необхідної точності та надійності. Нестабільні ринкові умови, динамічні зміни споживчих уподобань та конкурентних стратегій створюють величезні виклики. Традиційні методи часто не можуть врахувати новизну даних, що є вирішальним фактором точності та актуальності в сучасних умовах [1].

Основна увага в цій роботі зосереджена на розв'язанні проблеми розробки ефективного методу довгострокового прогнозування попиту, який враховує складні та динамічні зміни на ринку, а також новизну даних. Метою є створення моделі, яка буде не тільки точною і надійною, але й стане основою для прийняття стратегічних рішень в управлінні попитом і розвитку бізнесу.

Обрана методологія, заснована на використанні ансамблю нейронних мереж, демонструє високу потенційну гнучкість і здатність адаптуватися до складних нелінійних залежностей у даних про попит. Нейронні мережі полегшують моделювання взаємодії між різними змінними,

фіксуючи неявні взаємозв'язки. Використання ансамблю, де співпрацюють кілька нейронних мереж, допомагає уникнути перенавчання і підвищує загальну узагальненість моделі [2].

2. ЗАПРОПОНОВАНИЙ МЕТОД

Запропонований метод прогнозування базується на вдосконаленому методі пакування, який використовує ансамбль нейронних мереж. Для навчання базових моделей використовується адаптивний відбір даних на основі їхнього віку. Зазвичай, у звичайному методі пакування дані для кожної базової моделі вибираються випадковим чином з однаковою ймовірністю. Однак у нашому методі ми вирішили використовувати адаптивний відбір даних, коли старіші дані мають меншу ймовірність бути обраними для навчання.

Запропонований метод прогнозування базується на вдосконаленому методі пакування, який використовує ансамбль нейронних мереж. Для навчання базових моделей використовується адаптивний відбір даних на основі їхнього віку. Зазвичай, у звичайному методі пакування дані для кожної базової моделі вибираються випадковим чином з однаковою ймовірністю. Однак у нашому методі ми вирішили використовувати адаптивний відбір даних, коли старіші дані мають меншу ймовірність бути обраними для навчання.

Це досягається внаслідок використання функції вибору рядків, яка зменшує ймовірність вибору старих даних у навчальних підвбірках для базових моделей. Нехай ми маємо навчальну вибірку X , що складається з векторів x з n ознаками, $x = (x_1, x_2, \dots, x_n)$. І нехай серед цих ознак є часова ознака x_{date} для відповідного вектора x , оскільки ми розглядаємо дані з часовою залежністю. Ми використовуємо спадну функцію, яка дозволяє нам контролювати ймовірність відбору спостереження до підвбірки залежно від його віку, у вигляді:

$$\begin{aligned} \text{date}_{\min} &= \min_{x \in X} (x_{date}), \\ w &= e^{\frac{x_{date} - \text{date}_{\min}}{100}}, \\ p &= \frac{w}{\sum_{i=0}^n w_i}. \end{aligned}$$

Отримані значення ймовірності p для кожного спостереження x дозволяють збалансувати кількість старих спостережень у навчальній підвбірці та мати переважну більшість нових спостережень.

Ми реалізували цей метод як клас, який приймає ваги для кожного спостереження як вхідні параметри та конфігурацію базової моделі для пакування. Таким чином, можна налаштувати ваги для кожного спостереження і використовувати будь-який доступний алгоритм як базову модель, що забезпечує гнучкість у виборі моделей для ансамблю. На рисунку 1 показано загальну схему запропонованого методу.

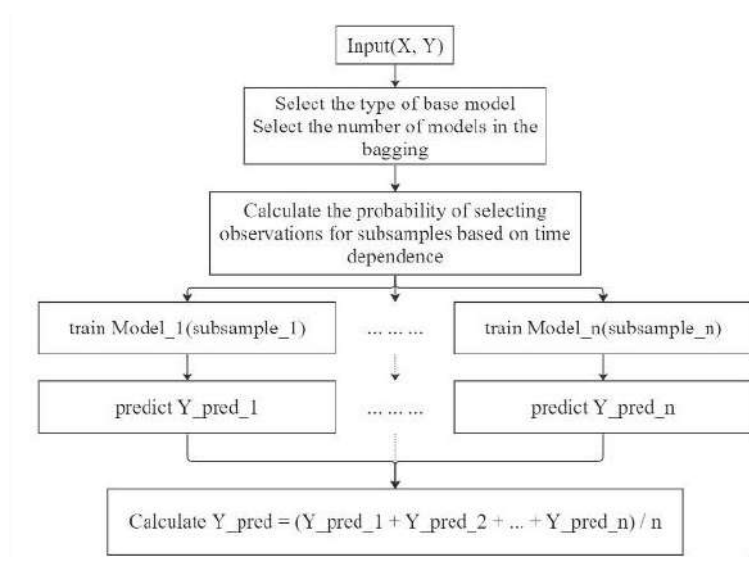


Рисунок 1. Загальна схема залежного від часу беггінгу

Як базову модель було обрано багатосаровий перцептрон з 25 прихованими шарами. Цей вибір зумовлений його здатністю працювати зі складними нелінійними залежностями в даних та високою гнучкістю у вираженні складних функцій.

3. РЕЗУЛЬТАТИ ТА ПОРІВНЯННЯ АЛГОРИТМІВ

3.1. Обробка даних

Для довгострокового прогнозування було обрано таку деталізацію даних: "адміністративний регіон країни" – "товар" – "тиждень". Горизонт прогнозування – 52 тижні. Прогнозування цільової змінної загального обсягу продажів. Приклад цільової змінної наведено на рисунку 2.

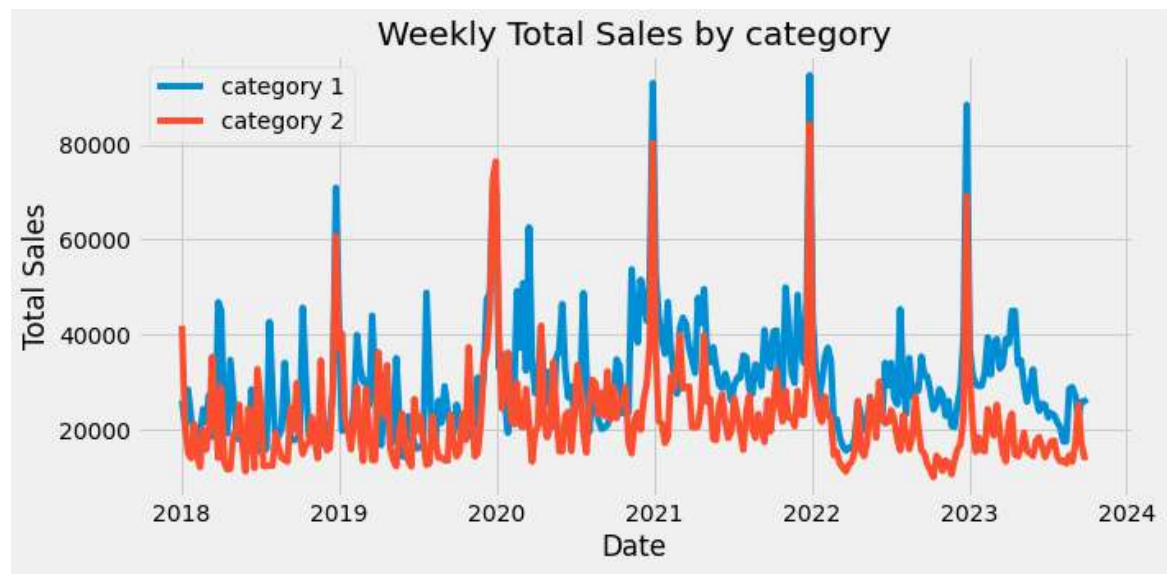


Рисунок 2. Приклад цільової змінної

Враховуючи важливість детального розгляду атрибутів у системі довгострокового прогнозування попиту для мережі магазинів, нижче наведено опис основних вхідних атрибутів:

- Загальна площа магазинів у певній місцевості може вказувати на місткість.

- Загальна кількість касових апаратів у магазинах певного регіону може свідчити про їхню пропускну здатність та пропускну спроможність.
- Дата початку тижня використовується для агрегування даних і визначення періоду аналізу.
- Кількість рекламних активностей за певний період для обраного продукту.
- Середня базова ціна продукту в регіоні.
- Середня акційна ціна товару в регіоні може бути ключовим фактором у визначенні вигоди для покупця.
- Опишіть характеристики товару: розмір, категорія, бренд.

Ці атрибути є важливими для нашого дослідження і дозволять нам краще зрозуміти та створити нові атрибути для прогнозування попиту на товари в мережі магазинів.

Після проведення тесту на стаціонарність було зроблено висновок, що ряд є стаціонарним

Щоб покращити якість алгоритмів, потрібно намагатися розробити або зібрати набір даних з найбільш репрезентативними характеристиками. Первинні дані не можуть одразу слугувати навчальними [3]. Було додатково створено наступні ознаки:

- Порядковий номер тижня в межах року.
- Послідовна кількість днів у році.
- Число місяця.
- Номер календарного кварталу.
- Циклічне представлення функцій дати для ілюстрації циклічної природи часу (наприклад, 12 місяців найближче до 1 місяця).
- Назва свята, яке було протягом цього тижня.
- Координати регіонального центру.
- Відстань від обласного центру до столиці.
- Населення регіону.
- Загальний обсяг продажів продукту за попередній період.
- Середні продажі продукції тієї ж категорії та розміру за попередній період.
- Індекс цін на основі середніх цін категорії.

Досліджувані дані містять набір категоріальних характеристик, які не можуть бути коректно перенесені в моделі машинного навчання без певних перетворень до числової форми. Для звичайних табличних даних можна розглядати більшість популярних методів кодування категоріальних змінних, таких як one-hot, label encoding, або методи з використанням цільової змінної, наприклад, target encoding тощо. Ось чому one-hot не відповідає нашим потребам, оскільки у нас багато категорій. Використання цього методу призведе до так званого прокляття розмірності. Також необхідно враховувати природу даних, а саме їхню залежність від часу, тобто при перетворенні слід використовувати лише минулу інформацію, щоб уникнути витoku даних. Для цього ми використаємо кодування за допомогою CatBoost. Алгоритм схожий на цільове кодування, але базується на принципі впорядкування для подолання проблеми заглядання в майбутнє. Він використовує принцип, схожий на перевірку даних часових рядів. Значення цільової статистики залежать від історії спостережень, тобто цільове значення для поточного об'єкта обчислюється тільки з серії спостережень перед ним [4].

Після цього дані були стандартизовані, щоб можна було використовувати алгоритми, які залежать від масштабу ознак, зокрема, різні варіації нейронних мереж.

3.2. Метрики для оцінки якості прогнозу

Для перевірки точності моделі прогнозування в цьому дослідженні використовуються такі методи оцінювання, як середня абсолютна похибка (MAE), зважене зміщення (wBias) та зважена середня абсолютна відсоткова похибка (wMAPE) [5]. Вони визначаються за наступними формулами:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_{true,i} - y_{pred,i}|$$

$$wBias = \frac{\sum_{i=1}^n (y_{true,i} - y_{pred,i}) \times weight_i}{\sum_{i=1}^n y_{true,i} \times weight_i}$$

$$wMAPE = \frac{\sum_{i=1}^n |y_{true,i} - y_{pred,i}| \times weight_i}{\sum_{i=1}^n y_{true,i} \times weight_i}$$

де $y_{true,i}$ – істинне значення i -го спостереження, $y_{pred,i}$ – прогнозоване значення для i -го спостереження, а $weight_i$ - реалізована ціна i -го спостереження.

3.3. Результати

У таблицях 1–3 наведено оцінки якості реалізованих алгоритмів пакування та інших алгоритмів на тестовій вибірці для різних категорій товарів.

Таблиця 1. Результати по тестовому зразку для категорії оливки

Алгоритм	MAE	wBias	wMAPE
Linear Regression	21,240999	0,171258	2,089721
Decision Tree	20,523671	0,153452	2,099562
KNN	14,37258	0,002988	1,464464
Light Gradient Boosting Machine	10,265612	0,016345	1,015855
MLP	10,196435	0,023553	0,983567
Запропонований беггінг: MLP	10,042824	0,016544	0,964849

Таблиця 2. Результати по тестовому зразку для категорії вода

Алгоритм	MAE	wBias	wMAPE
Linear Regression	84,480005	0,425613	1,390737
Decision Tree	74,053743	0,685274	0,687261
KNN	69,532563	0,22464	0,265432
Light Gradient Boosting Machine	66,621567	0,11069	0,228842
MLP	65,466542	0,14347	0,202356
Запропонований беггінг: MLP	65,023453	0,10353	0,188464

Таблиця 3. Результати по тестовому зразку для категорії сир

Алгоритм	MAE	wBias	wMAPE
Linear Regression	22,41161	0,583439	1,174249
Decision Tree	26,370057	0,688346	1,373164
KNN	15,777434	0,315182	0,819989
Light Gradient Boosting Machine	9,615076	0,113314	0,494714
MLP	9,605615	0,146724	0,482369
Запропонований беггінг: MLP	9,589425	0,127032	0,482391

Погляньмо на агреговані прогнози для розробленого алгоритму, які показані на рисунку 3 (тижні по осі X, загальні продажі по осі Y). Ми бачимо, що модель в цілому досить добре відстежує зміни рівня продажів, що є критичним при створенні моделі, і немає сильного зміщення.

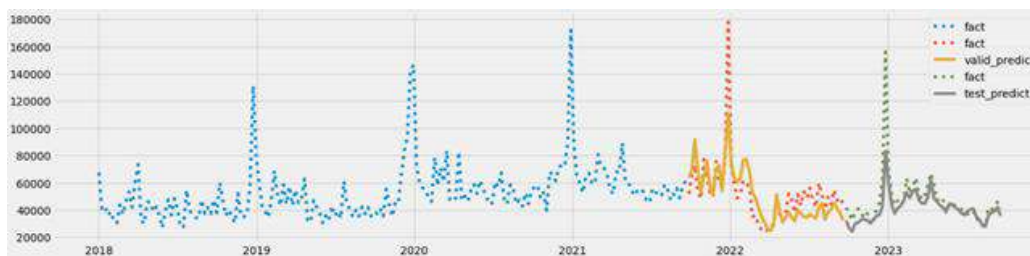


Рисунок 3. Прогнозування на валідаційній та тестовій вибірці

4. ВИСНОВКИ

У цій науковій роботі ми представляємо новий метод довгострокового прогнозування попиту на основі ансамблю нейронних мереж, який враховує новизну даних. Наш підхід використовує метод мішків і техніку модифікації, що дозволяє враховувати актуальність і новизну даних при генерації навчальних вибірок для кожної моделі в ансамблі.

Результати наших експериментів показали, що запропонований метод перевершує наявні підходи до довгострокового прогнозування попиту. Результати підкреслюють важливість врахування новизни даних при розробці стратегій прогнозування попиту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aaltonen K. Project stakeholder analysis as an environmental interpretation process. *International journal of project management*. 2011. Т. 29, № 2. С. 165–183. URL: <https://doi.org/10.1016/j.ijproman.2010.02.001> (дата звернення: 20.11.2023).

2. Perin G., Chmielewski Ł., Picek S. Strength in numbers: improving generalization with ensembles in machine learning-based profiled side-channel analysis. *IACR transactions on cryptographic hardware and embedded systems*. 2020. С. 337–364. URL: <https://doi.org/10.46586/tches.v2020.i4.337-364> (дата звернення: 20.11.2023).

3. The uncertainty estimation of feature-based forecast combinations / X. Wang та ін. *Journal of the operational research society*. 2021. С. 1–15. URL: <https://doi.org/10.1080/01605682.2021.1880297> (дата звернення: 20.11.2023).

4. Sales forecasting based on catboost / J. Ding та ін. 2020 2nd international conference on information technology and computer application (ITCA), м. Guangzhou, China, 18–20 груд. 2020 р. 2020. URL: <https://doi.org/10.1109/itca52113.2020.00138> (дата звернення: 20.11.2023).

5. Hyndman R. J., Athanasopoulos G. *Forecasting: principles and practice*. OTexts, 2021

ПОШУК ВІДПОВІДНОСТІ МІЖ ЗОБРАЖЕННЯМ І ЙОГО ТЕКСТОВИМ ОПИСОМ

Шаповал Н.В.¹, Крижанівська О.В.²

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

¹ shovgun@gmail.com, ² kruzhanivska.olena212@gmail.com

В сучасному інформаційному світі, коли обсяги доступної інформації надзвичайно великі, виникає необхідність розробки та вдосконалення методів автоматичної обробки текстової і графічної інформації. Однією з актуальних задач є встановлення відповідності між зображеннями і мовними описами, що відкриває безліч можливостей для розвитку різних застосувань, включаючи покращення пошукових систем, автоматичне розпізнавання зображень та підтримку людей з обмеженими можливостями. Відповідність між графічною та текстовою інформацією важлива для багатьох сучасних додатків і систем, таких як пошукові системи, розпізнавання образів, аналіз медичних даних і багато інших. Метою дослідження є запропонувати модифіковану модель такого типу. Це дозволить покращити якість та швидкість роботи інформаційних систем, що базуються на аналізі інформації.

Ключові слова: мультимодальна обробка інформації, Image-Text matching, автоматична обробка текстової та графічної інформації, модифікація нейронних мереж, моделі розпізнавання образів.

1. ВСТУП

Сучасна епоха інформаційних технологій та глибокого навчання свідчить про стрімкий розвиток інформаційних систем та обробки даних. Існує потреба у моделях, які дозволяють ефективно працювати з інформацією, незалежно від її формату. Окрім того, в сучасному інформаційному світі, коли обсяги доступної інформації надзвичайно великі, виникає необхідність розробки та вдосконалення методів автоматичної обробки текстової і графічної інформації. Проте суспільство стикається зі значним обсягом інформації, представленої в різних форматах, включаючи текст, зображення та відео. Здійснення ефективної обробки та аналізу цих різних видів інформації стає важливою задачею в контексті пошукових систем, автоматизованих систем ідентифікації, медицини, автономної навігації, відеоспостереження, робототехніки, створення інтерактивних мультимедійних додатків, інформаційної безпеки та інших областях. Однією з актуальних задач є встановлення відповідності між зображеннями і мовними описами, що відкриває безліч можливостей для розвитку різних застосувань, включаючи покращення пошукових систем, автоматичне розпізнавання зображень та підтримку людей з обмеженими можливостями. Знаходження взаємозв'язків між зображеннями та текстовим описом має широкі застосування, починаючи від систем пошуку та індексації великих обсягів мультимедійних даних до розвитку систем розпізнавання образів та інтелектуального пошуку в онлайн-ресурсах. Ця проблема стає критичною в таких галузях, як електронна комерція, медична діагностика, туризм та соціальні мережі, де точне розуміння контенту, представленого зображеннями та текстовими описами, може значно поліпшити користувацький досвід та ефективність рішень. Отже, розробка методів та алгоритмів, які

забезпечують високий рівень відповідності між зображеннями та текстовим описом, є актуальною задачею.

2. IMAGE-TEXT MATCHING МОДЕЛІ

Існує багато нейронних мереж, які дозволяють вирішити дану задачу. В даному дослідженні були розглянуті нижче приведені моделі.

SAM. Модель SAM – це мультимодальна модель, яка вирішує завдання швидкої сегментації на основі текстових підказок і може бути адаптована під задачу Image-Text matching [1]. Задачею моделі є сегментація зображення, семантична сегментація, візуальна підказка. Набір даних, на якому навчалась модель SA-1B [1]. Він складається з 11 мільйонів різноманітних зображень з високою роздільною здатністю, ліцензованих і захищених від несанкціонованого доступу, а також 1,1 мільярда високоякісних масок сегментації, зібраних за допомогою нашого рушія даних. Що відносно архітектури, то завдання швидкої сегментації та мета реального використання накладають обмеження на неї. Зокрема, модель повинна підтримувати гнучкі підказки, обчислювати маски в амортизованому режимі реального часу, щоб забезпечити інтерактивне використання, і враховувати неоднозначність. Виявлено, що простий дизайн задовольняє усім трьома обмеженнями: потужний кодер зображень обчислює вбудовування зображень, кодер підказок вбудовує підказки, а потім ці два джерела інформації об'єднуються в легкому декодері масок, який прогнозує маски сегментації (Рис. 1) [1].

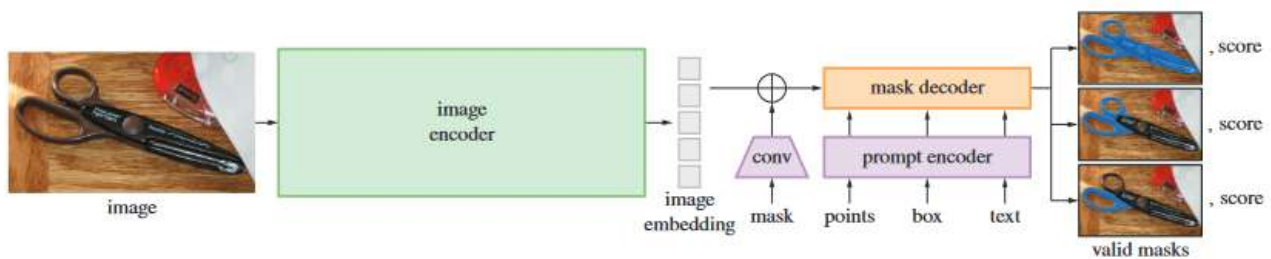


Рисунок 1. Архітектура моделі SAM

Переваги та недоліки:

- немає вбудованих зображень;
- немає підтримки української мови;
- проект не open source;
- + модель навчена на багатьох датасетах;
- + достатня точність.

EVA. Модель EVA використовується для багатьох задач, таких як детекція об'єктів, семантичної сегментації, класифікації зображень, тощо. Мультимодальна модель, яка вводить зображення та одну або декілька підказок і виводить правильну маску сегментації. Окрім цього, слід зауважити, що SAM – state of the art модель [2]. В своїй основі вона використовує модель CLIP. EVA – це Vanilla ViT, попередньо навчений реконструювати замасковані зображення-текст, вирівняні за візуальними ознаками, на основі видимих ділянок зображення. EVA масштабована до одного мільярда параметрів і встановила певні рекорди в широкому спектрі репрезентативних задач, таких як розпізнавання зображень, розпізнавання відеодій, виявлення об'єктів, сегментація екземплярів. При навчанні було використано в якості набору даних CC12M і CC3M (дані зображень без підписів), COCO і ADE20K (дані навчальних наборів), ImageNet-21K і Object365 (дані зображень) [2]. Об'єднаний набір даних для попереднього навчання містить загалом 29,6 мільйонів зображень.

Переваги та недоліки:

- низька точність (класифікація 75,7%, детекція та сегментація 53,4%) [2];
- важко інсталювати та використовувати;
- наразі багато багів;
- немає підтримки української мови;
- + проект open source.

GLIDE. GLIDE модель - це генеративна модель на основі стабільної дифузії призначена для генерації тексту в зображення та навпаки з тексту в зображення [3]. Також використовується модель CLIP. Завдання: генерація зображень, зафарбовування зображень, генерація тексту в зображення, генерація тексту в зображення з нульового кадру. Використаний набір даних – COCO [3]. Модель абляційної дифузії (ADM), навчена генерувати зображення 64 x 64 (ImageNet), текстова модель (трансформатор), яка впливає на генерацію зображення за допомогою текстової підказки, і модель апсемплінгу, яка перетворює наші маленькі зображення 64 x 64 на більш зрозумілі 256 x 256 пікселів [3]. Перші два компоненти взаємодіють один з одним, щоб керувати процесом генерації зображень для точного відображення текстової підказки, а останній необхідний для того, щоб зробити зображення, які ми створюємо, більш легкими для інтерпретації.

Переваги та недоліки:

- немає вбудованих зображень;
- на виході отримуємо зображення лише у розмірі 256px;
- модель навчена лише на одному датасеті;
- проект не open source;
- + є підтримка української мови.

CLIP. CLIP (Contrastive Language-Image Pre-Training) – це мультимодальна модель, яка вивчає відповідність між природною мовою та зображеннями [4]. Завдання моделі наступні: розпізнавання дій, бенчмаркінг, класифікація зображень, класифікація зображень, навчання з довгим хвостом, класифікація мемів, розуміння природної мови, категоризація об'єктів, виявлення атрибутів відкритого словника, узагальнення поза розподілом, швидка інженерія, класифікація зображень під наглядом, генерація тексту, крос-модальний пошук з нульовим результатом, навчання з нульовим результатом, класифікація зображень з нульовою передачею. У ході навчання мережі було використано велику кількість наборів даних, серед них ImageNet, COCO, KITTI, GLUE, SST, UCF101, Visual Genome, Flickr30k, SUN397 та 22 інші [4]. Відносно архітектури, автори використали дві різні основи (Resnet50 і Vision Transformer (ViT)) для кодера зображень і трансформатор як основу для кодера тексту (Рис. 2) [4].

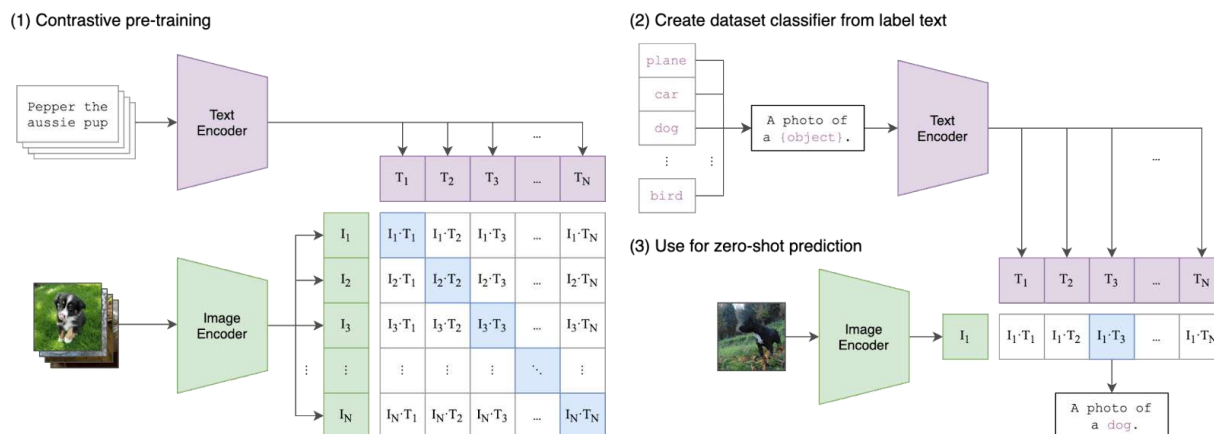


Рисунок 2. Модель CLIP

Переваги та недоліки:

- немає підтримки української мови;
- низька точність (image retrieval flickr30k 68,7%, mscoco 37,8%, text retrieval flickr30k 88%, mscoco 58,1%);
- + є вбудовані зображення;
- + модель навчена на багатьох датасетах;
- + легко інсталиувати та використовувати.

Аналізуючи ці рішення, було вирішено використовувати модель CLIP, оскільки вона є базою для інших рішень і може потім бути адаптована під інші задачі. В моделі CLIP завдання полягає в пошуку максимально близького текстового опису для зображень. Навчена модель здатна розпізнавати предмети, навіть якщо вони представлені в зовсім іншому контексті. CLIP містить дві мережі. Одна мережа повинна обробляти зображення та створювати векторну репрезентацію на основі цих зображень, тоді як інша мережа має обробляти текст та створювати векторну репрезентацію тієї ж розмірності, що й перша мережа. Оскільки було б неможливо вибрати дві нейронні мережі з різних галузей (CV і NLP), які мали б однакові вихідні характеристики, до обраної архітектури слід додати шар, який приведе їхні розмірності до єдиної величини. Ці шари можна назвати лінійними перетвореннями для забезпечення відповідності розмірностей.

3. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

У ході дослідження була використана модель CLIP як основа для вивчення ключових показників. В якості датасету використано власноруч українізований набір даних Flickr з 30 тисячами екземплярів тренувальних та валідаційних пар зображення та текстового опису. Початкові результати за перші чотири епохи тренування показали втрати на рівні 7,8% під час тренування та 9,4% під час валідації. Ці показники слугували вихідною точкою для подальших оптимізацій та вдосконалень. Для підвищення ефективності моделі, було внесено зміни у процес тренування. Оптимізатор Adam був замінений на AdamW, а потім AdamW був замінений на AdaBelief, що включає адаптивне відсікання градієнту, а також додано планувальник темпу навчання з подальшим коригуванням гіпер-параметрів всієї мережі. Планувальник темпу навчання в даному випадку грає роль механізму, який визначає, як швидко параметри моделі оновлюються під час процесу тренування. Темп навчання визначає величину кроку або кроків, які використовуються для оновлення ваг моделі під час зворотного поширення під час тренування. Роль планувальника темпу навчання полягає в оптимізації

процесу тренування, забезпеченні стійкості та швидкості збіжності моделі. Після цих модифікацій знову було проведено тренування і нові результати вже за перші чотири епохи показали помітне покращення: втрати під час тренування зменшилися до 1,14%, а втрати під час валідації склали 2,28%. Більш детальна інформація наведена нижче (Табл. 1).

Таблиця 1. Порівняння втрат моделей без модифікацій та з модифікаціями

CLIP+Adam		CLIP+AdamW		CLIP+AdaBelief+ACG	
втрати під час тренування	втрати під час валідації	втрати під час тренування	втрати під час валідації	втрати під час тренування	втрати під час валідації
33,3	31,3	3,12	2,67	2,58	2,38
14,03	15,4	2,05	2,63	1,51	2,3
9,2	10,9	1,81	2,6	1,27	2,29
7,8	9,4	1,68	2,57	1,14	2,28

4. ВИСНОВОК

Експеримент демонструє, що заміна оптимізатора на AdaBelief з адаптивним відсіканням градієнту та додавання планувальника темпу навчання з подальшим коригуванням гіперпараметрів, суттєво покращили продуктивність моделі CLIP. Зменшення втрат під час тренування та валідації свідчить про ефективність нових стратегій оптимізації та навчання. Це не лише підкреслює значущість правильного вибору оптимізаційних методів, але й вказує на потенційні можливості подальшого вдосконалення моделей для вирішення складних завдань.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alexander Kirillov: Segment Anything [Електронний ресурс] – Режим доступу: <https://arxiv.org/pdf/2304.02643.pdf>
2. Yuxin Fang: EVA: Exploring the Limits of Masked Visual Representation Learning at Scale [Електронний ресурс] – Режим доступу: <https://arxiv.org/pdf/2211.07636.pdf>
3. Alex Nichol: GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models [Електронний ресурс] – Режим доступу: <https://arxiv.org/pdf/2112.10741.pdf>
4. Alec Radford: Learning Transferable Visual Models From Natural Language Supervision [Електронний ресурс] – Режим доступу: <https://arxiv.org/pdf/2103.00020.pdf>