КПІ ім. Ігоря Сікорського

Матеріали IV Міжнародної науково-практичної конференції
молодих вчених та студентів
«Інженерія програмного забезпечення
і передові інформаційні технології (SoftTech-2023)»
присвяченої 125-й річниці КПІ ім. Ігоря Сікорського

SoftTech-2023

весна

2021

9-11 травня
Україна, Київ

УДК: 004.4

Матеріали IV Міжнародної науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2023). Матеріали конференції. – Київ. – 2023. 9-11 травня 2023 р. – 93 с.

Збірник містить тези доповідей, що були представлені на міжнародній науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023)», присвяченій 125-й річниці КПІ ім. Ігоря Сікорського. В доповідях розглянуті сучасні наукові та практичні проблеми інформатики та програмної інженерії.

Конференцію зареєстровано в Українському інституті науково-технічної експертизи та інформації. Посвідчення про реєстрацію № 205.

Дизайн обкладинки
Майєр З.О., провідний інженер кафедри інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра ІПІ ФІОТ, КПІ ім. Ігоря Сікорського, 2023

СКЛАД ПРОГРАМНОГО КОМІТЕТУ

Голови:

д.т.н., проф. Корнага Я.І. – декан факультету інформатики та обчислювальної техніки;

д.т.н., проф. Жаріков Е.В. – завідувач кафедри інформатики та програмної інженерії.

Члени програмного комітету:

д.т.н., професор Вольтер Клаус-Юрген – директор Інституту технологій комплектації та з'єднання електроніки; заступник директора Фраунгофер інституту керамічних технологій і систем ІКТС діагностики матеріалів МД (ІКТС-МД), Технічний університет Дрездена, м. Дрезден, Німеччина;

д.т.н., професор Любомир Ванков Димитров – декан машинобудівного факультету, Технічний університет, м.Софія, Болгарія;

к.т.н., доцент Ху Чженбін – Педагогічний університет Центрального Китаю, м. Ухань, Китай;

Леонід Кун – CEO at Abona Deutschland GmbH, м. Брухзаль, Німеччина;

д.т.н., професор Снитюк В.Є. – декан факультету інформаційних технологій, Київський національний університет імені Тараса Шевченка;

д.т.н., професор Купін А.І. – завідувач кафедри комп'ютерних систем та мереж, Криворізький національний університет;

д.т.н., професор Чалий С.Ф. – професор кафедри інформаційних управляючих систем, Харківський національний університет радіоелектроніки;

д.т.н., професор Гнатушенко В.В. – професор кафедри інформаційних систем та технологій, Національна металургійна академія України;

д.т.н., професор Бабічєв С.А. – професор кафедри фізики та методики її навчання, Херсонський державний університет;

д.т.н., професор Литвиненко В.І. – завідувач кафедри інформатики і комп'ютерних наук, Херсонський національний технічний університет;

д.т.н., професор Рудакова Г.В. – професор кафедри автоматизації, робототехніки і мехатроніки, Херсонський національний технічний університет;

д.т.н., професор Павлов О.А. – професор кафедри інформатики та програмної інженерії;

д.т.н., професор Стеценко І.В. – професор кафедри інформатики та програмної інженерії;

д.т.н., професор Сидоров М.О. – професор кафедри інформатики та програмної інженерії;

к.т.н., доцент Фіногенов О.Д. – доцент кафедри інформатики та програмної інженерії;

к.т.н., доцент Лісовиченко О.І. – доцент кафедри інформатики та програмної інженерії;

к.т.н., доцент Ліхоузова Т.А. – доцент кафедри інформатики та програмної інженерії.

СКЛАД ОРГАНІЗАЦІЙНОГО КОМІТЕТУ

Голова:

д.т.н., проф. Жаріков Е.В. – завідувач кафедри інформатики та програмної інженерії.

Члени організаційного комітету:

к.т.н., доцент Муха І.П. – доцент кафедри інформатики та програмної інженерії;
к.т.н. Ліщук К.І. – доцент кафедри інформатики та програмної інженерії;
к.т.н. Олійник Ю.О. – доцент кафедри інформатики та програмної інженерії;
к.т.н., доцент Баклан І.В. – доцент кафедри інформатики та програмної інженерії;
к.т.н., доцент Гавриленко О.В. – доцент кафедри інформаційний систем та технологій;
Халус О.А. – ст. викл. кафедри інформатики та програмної інженерії;
Лукутін О.В. – ст. викл. кафедри інформатики та програмної інженерії;
к.т.н. Сирота О.П. – доцент кафедри інформатики та програмної інженерії;
к.т.н., доцент Іванова Л.М. – доцент кафедри інформатики та програмної інженерії;
к.т.н., доцент Крамар Ю.М. – доцент кафедри інформатики та програмної інженерії;
к.е.н. Родіонов П.Ю. – доцент кафедри інформатики та програмної інженерії;
Марченко О.І. – ст. викл. кафедри інформатики та програмної інженерії;
Ковтунець О.В. – ст. викл. кафедри інформатики та програмної інженерії;
Вітковська І.І. – ст. викл. кафедри інформатики та програмної інженерії;
Очеретяний О.К. – асистент кафедри інформатики та програмної інженерії.

ЗМІСТ

УДК 004.412

Кошовець Євгеній Павлович, *здобувач вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник: Жаріков Едуард В'ячеславович,
доктор технічних наук, професор, завідувач кафедри інформатики та програмної інженерії
КПІ ім. Ігоря Сікорського, Україна

Е-КАТАЛОГ МЕТРИК ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
E-CATALOGUE FOR METRICS OF SOFTWARE DEVELOPMENT LIFECYCLE

Анотація. Метрика програмного забезпечення — це кількісний показник, який використовується для оцінки різних аспектів розробки програмного забезпечення, таких як якість, складність і продуктивність. Показники можна використовувати для оцінки ефективності процесів розробки програмного забезпечення, визначення областей покращення та відстеження прогресу з часом. Вибір метрик для використання сьогодні не є тривіальним завданням через багато альтернатив і відсутність загальнодоступного каталогу з визначеною структурою. У статті проаналізовано існуючі спроби створення каталогу у спосіб визначення певного формату або шаблону. У результаті аналізу запропоновано більш ефективний електронний каталог.

КЛЮЧОВІ СЛОВА: МЕТРИКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЖИТТЄВИЙ ЦИКЛ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, КАТАЛОГ.

Abstract. A software metric is a quantitative measure that is used to evaluate various aspects of software development, such as quality, productivity, complexity, and performance. Metrics can be used to assess the effectiveness of software development processes, identify areas of improvement, and track progress over time. Choosing a metric to use these days is not a trivial task, because of a lot of alternatives and not having a shared repository with defined structure. In this publication, existing tries of creating a catalogue by defining a certain format or template were described. On this basis a newer and more comfortable e-catalogue was proposed.

KEYWORDS: SOFTWARE METRIC, SOFTWARE DEVELOPMENT LIFECYCLE, CATALOGUE

Вступ. Метрика програмного забезпечення — це кількісний показник, який використовується для оцінки різних аспектів розробки програмного забезпечення, таких як якість, складність і продуктивність. Метрики можуть бути корисними у таких аспектах:

- вимірювання прогресу: метрики дозволяють вимірювати хід проекту розробки програмного забезпечення;
- покращення якості: метрики дозволяють виявляти проблеми з якістю в процесі розробки програмного забезпечення;
- підвищення продуктивності: метрики можна використовувати для вимірювання продуктивності та визначення можливостей для покращення;
- прийняття рішень на основі даних: метрики надають об'єктивні дані, які можна використовувати для прийняття обґрунтованих рішень;
- полегшення спілкування: метрики можна використовувати для покращення взаємодії учасників групи розробників програмного забезпечення.

Отже, враховуючи вищеописані переваги можна зазначити, що використання метрик під час розробки програмного забезпечення є невід'ємною частиною цього процесу на всіх етапах його життєвого циклу. На жаль,

сьогодні не існує єдиного репозиторію всіх необхідних метрик. Через це пошук релевантних метрик стає надважким і займає багато часу. Вирішити дану проблему може каталог метрик. Додавання певних метрик до каталогу є важливою процедурою, але це неможливо зробити без заздалегіть продуманого формату та структури каталогу. Тому розроблення структури, формату та складу метрик програмного забезпечення є актуальною задачею.

Аналіз публікацій. Огляд літератури проводився із використанням Systematic mapping study [1]. Дослідницькі питання стосуються уніфікації, класифікації метрик та існування шаблонів чи форматів їх представлення. У результаті аналізу літератури отримано 18 публікацій. Більшість із них мали кінцевою метою саме визначити набір метрик, який би був релевантний у тому чи іншому проєкті. У статті [2] представлений аналіз метрик для застосунків, які використовують модель IaaS [3]. Метрики безпеки для хмарних застосувань проаналізовано у статті [4]. У статті [5] запропоновано представлення метрики у вигляді шаблону з прив'язкою до мов програмування. На жаль, представлення такого шаблону є доволі простим і загальним, тому не несе цінності. Розглянуто також декілька робіт, які вводять формат представлення метрик [6, 7]. Рішення, представлені у [6] є не актуальними для сучасних проєктів. Наприклад, в шаблоні є поле «тип збору даних», значення якого автоматичний або мануальний. Також цей шаблон є спеціалізованим за призначенням і створювався для конкретного фреймворку для збору метрик веб-застосувань. У статті [7] запропоновано формат узагальненого каталогу метрик. Автори [8] сформулювали питання, відповівши на які можно вважати метрику визначеною. При цьому автори змогли покрити 7 із 10 метрик. Не дивлячись на те, що формату достатньо, щоб повно описати метрику, деякі поля мають дуже глибоке підґрунтя, при цьому тільки заплутуючи користувача, а не допомагаючи вибрати метрику. Наприклад, поле «валідність», яке збирає в собі результати різних дослідів, проведених над метрикою. Також, відсутність розбиття на класи в тому чи іншому вигляді не спрощує сприйняття та розуміння сфери застосування метрик.

Основна частина. У результаті аналізу публікацій встановлено, що процес розроблення структури, формату та складу метрик програмного забезпечення треба покращувати, в тому числі і у вигляді каталогу.

Аналогічно пропозиціям у [6] і [7], візьмемо за основу деякі поля, які є актуальними для багатьох проєктів розробки програмного забезпечення: назва, ціль, сутність та її тип, атрибут та його тип, рівень метрики, шкала та одиниці вимірювання, діапазон можливих значень. Ці важливі поля, які успішно використовувались у попередніх роботах, є корисними і зрозумілими. Але деякий опис потребує доопрацювання. Немає сенсу тримати поле «формула», якщо дуже часто вона буде або відсутня (для базових метрик), або буде мати складний вигляд, який без додаткових відомостей не буде зрозумілим. Пропонується замінити його полем «визначення», в якому буде зберігатися за наявності формула чи алгоритм обчислення метрики. Якщо у формулі присутні інші метрики, вони мають бути у вигляді посилань. Пропонується також замінити поле «метрики до яких відноситься» на «метрики, для яких є базовою», тобто зберігатимемо метрики на рівень вище, адже ті, які на рівень нижче, уже є у визначенні. Доцільно розширити поле «інструменти» і додати тип автоматизації (наприклад збір інформації, візуалізація результатів), а також теги для відкидання неактуальних варіантів. Нові поля: аліаси (англ. Alias), які містять альтернативні назви або скорочені назви; наслідки, які містять позитивні та негативні описи при досягненні або недосягненні оптимальних значень; критичні точки (зберігають показники, при яких отримане значення можна вважати оптимальним, добрим або поганим); фази життєвого циклу (містять фази життєвого циклу, при яких

застосовується); категорія та підкатегорії метрик, які були взяті із стандарту «ISO/IEC 25010» [9]; нормалізація (описує, в яких випадках і як можна нормалізувати метрику для більш точного обчислення), обмеження (описує слабкі місця метрики, випадки, коли застосування нерелевантне); можливі використання (описує сценарії, які доречно розглянути до застосування програмного забезпечення).

Отже, пропонується формат каталогу, який представимо на прикладі відомої метрики «кількість рядків коду»:

- назва : lines of code;
- аліаси: LOC, code size;
- ціль: виміряти розмір і складність програмного забезпечення, підрахувавши кількість рядків коду;

наслідки:

- хороші наслідки: більше значення вказує на складнішу програму, що може продемонструвати зусилля та ресурси, вкладені в розробку програмного забезпечення;
- погані наслідки: велике значення може ускладнити підтримку програми та збільшити ймовірність помилок. Це також може призвести до більш довгих циклів розробки та тестування;
- сутність: програма;
- тип сутності : продукт;
- атрибут: розмір;
- тип атрибута: внутрішній;
- фази життєвого циклу: розробка;
- рівень метрики: базова;
- категорія метрики: продуктивність;
- підкатегорія метрики: використання ресурсів;
- шкала вимірювання: абсолютна;
- одиниці вимірювання: кількість рядків коду;
- критичні точки:
- оптимально: 1000 < рядки маленького проекту < 10000, 5000 < рядки великого проекту < 50000;
- добре: менше ніж 1000 рядків для маленьких проектів, до 5000 рядків для більших проектів;
- погано: більше 10000 рядків для малих проектів, та більше 50000 для великих проектів;
- визначення: підраховує кількість рядків у програмі;
- метрики, в яких використовується: цикломатична складність, MI;
- інструменти:
- назва: CLOC;
- тип автоматизації: збирає дані;
- теги: Java, Python, C++;
- можливі використання:
- оцінка складності коду та оцінка витрат на розроблення та підтримку;
- порівняння розміру коду різних версій однієї програми;
- обмеження:
- не враховує якість чи читабельність коду;
- на LOC може впливати використання бібліотек або засобів генерації коду;
- може відрізнятися залежно від мови програмування та стилю кодування, що використовується;
- нормалізація: нормалізацію можна використовувати для врахування відмінностей у мовах програмування або характері проекту. Наприклад, коефіцієнт нормалізації може бути застосований для коригування відмінностей у синтаксисі або структурних вимогах між мовами;
- аналіз тренду: аналіз тенденцій є стабільним для цього показника протягом тривалого часу, але слід бути обережним, порівнюючи розмір коду в різних проектах або мовах програмування.

Додаткові поля надають більшої гнучкості процесу фільтрації метрик при пошуку. Також вони зберігають актуальну інформацію, наприклад, можна побачити, який інструмент підходить для розробки, а який ні. Також додані нові поля, такі як «наслідки» та «ліміти», які дозволяють відповісти на питання із публікації [8].

Висновки. У дослідженні за допомогою systematic mapping study розглянуто поточний стан уніфікації метрик програмного забезпечення. Проаналізовано декілька існуючих рішень шаблонів метрик. На їх основі запропонована покращена структура електронного каталогу метрик. У результаті, формат каталогу містить більше полів, які орієнтовані на швидкий пошук та вибір потрібної метрики.

Список інформаційних джерел

1. What is Systematic Mapping Study. – Режим доступу: https://www.igi-global.com/dictionary/systematic-mapping-study/76946
2. Dalla Palma, S., Di Nucci, D., Palomba, F., & Tamburri, D. A. (2020). Toward a catalog of software quality metrics for infrastructure code. Journal of Systems and Software, 170, 110726..
3. What Is IaaS? Infrastructure as a Service Explained – Режим доступу: https://www.emnify.com/iot-glossary/iaas
4. Casola, V., De Benedictis, A., Rak, M., & Villano, U. (2018). A security metric catalogue for cloud applications. In Complex, Intelligent, and Software Intensive Systems: Proceedings of the 11th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2017) (pp. 854-863). Springer International Publishing.
5. Ducasse, S., Denier, S., Balmas, F., Bergel, A., Laval, J., & Mordal-Manet, K. (1998). Software metric for Java and C++ practices. Communication of the ACM, 41(12), 73-78..
6. Olsina, L., González Rodríguez, J., Lafuente, G. J., & Pastor, O. (2001). Towards Automated Web Metrics. In VIII Quality Brazilian Workshop, RJ-Br (pp. 74-86)..
7. Bouwers, E., Deursen, A. V., & Visser, J. (2014, June). Towards a catalog format for software metrics. In Proceedings of the 5th international workshop on emerging trends in software metrics (pp. 44-47).
8. Kaner, C. (2004). Software engineering metrics: What do they measure and how do we know?. In Proc. Int'l Software Metrics Symposium, Chicago, IL, USA, Sept. 2004 (pp. 1-12).
9. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. (2011). ISO/IEC 25010: Systems and software engineering-systems and Software Quality Requirements and Evaluation (SQuaRE). 2011.

УДК 004.4`24

Нестеренко Костянтин Павлович, *здобувач вищої освіти другого (магістерського) рівня КПІ ім. Ігоря Сікорського, Україна*
Науковий керівник: Стеценко Інна Вячеславівна, *д.т.н., професор, професор кафедри інформатики та програмної інженерії НТУУ "КПІ ім. Ігоря Сікорського", Україна*

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ КОНВЕРТАЦІЇ ЗОБРАЖЕНЬ У ТЕКСТУРИ

Анотація. У даному науковому дослідженні проведено аналіз предметної області використання програмного забезпечення для конвертації зображень у текстури. Сформовано основні вимоги до подібного програмного забезпечення, проаналізовано існуючі у вільному доступі рішення, виокремлені їх переваги та недоліки. Також розроблено власне рішення метою якого є покращення швидкодії відносно існуючих програмних засобів.
КЛЮЧОВІ СЛОВА: ТЕКСТУРА, DDS, БАГАТОПОТОЧНІСТЬ, ПРОГРАМНИЙ ЗАСІБ, АЛГОРИТМ БЛОЧНОГО СТИСНЕННСЯ ТЕКСТУР.

Abstract. This scientific research provides an analysis of the subject area related to the use of software for converting images into textures. The main requirements for such software are formulated, and existing freely available solutions are analyzed, identifying their advantages and disadvantages. Additionally, a proprietary solution is developed with the aim of improving speed compared to existing software tools.
KEY WORDS: TEXTURE, DDS, MULTITHREADING, SOFTWARE, TEXTURE BLOCK COMPRESSION ALGORITHM.

Вступ. З розвитком технологій все більш різноманітне програмне забезпечення інтегрується у різні сфери життєдіяльності людини та стає невід'ємною їх складовою. Так, програмне забезпечення з використанням двовимірної або тривимірної графіки, яке на початку використовувалося здебільшого у сфері розваг, активно використовується в медицині, військово-промисловому комплексі та інших галузях.
Проте, усе це програмне забезпечення в своїй роботі спирається на використання графічного процесору, який зі свого боку є пристроєм оптимізованим для роботи з певними форматами даних. Якщо ми говоримо про зображення, то таким форматом будуть DDS текстури [1]. Даний формат є найбільш вживаним, оскільки окрім того що виник одним з перших, дозволяє суттєво зменшити розмір зображення в пам'яті, в середньому в 4-6 рази, в залежності від алгоритму конвертації. Тобто будь-яке програмне забезпечення з двовимірною або тривимірною графікою так чи інакше потребує конвертації зображень, представлених у форматі JPEG, PNG, BMP тощо, у формат даних DDS.
Конвертація зображення у DDS текстуру це досить довготривала операція, оскільки вона вимагає великої кількості математичних операцій. А, оскільки з часом вимоги до якості комп'ютерної графіки постійно зростали, оригінальні зображення можуть мати досить великий розмір. І, коли мова йде про сотні, а часто й тисячі таких зображень, швидкодія стає однією з основних вимог до програмних засобів для конвертації зображень у DDS текстури.
Метою даного дослідження є підвищення швидкодії конвертації зображень у DDS текстури. За результатами аналізу існуючих рішень, оцінкою їх швидкодії та придатністю до використання з великою кількістю вхідних зображень розроблено новий програмний засіб для конвертації зображень у DDS текстури з більшою швидкодією.

Предметна область та формування вимог до програмного засобу. Як було зазначено раніше, потреба у конвертації зображень у DDS присутня фактично для будь-якого програмного забезпечення яке використовує двовимірну або тривимірну графіку, без прив'язки до конкретної області застосування. Проте вимоги для подібного програмного забезпечення є спільними. Насамперед воно повинно бути здатним обробляти великі масиви зображень. Для більшості проєктів мова йде про сотні зображень, якщо не тисячі. При цьому зображення можуть регулярно оновлюватися, і відповідно, їх потрібно конвертувати заново. Не менш важливою є вимога до швидкодії подібного програмного забезпечення. Через усе вище зазначене, повільний програмний засіб буде витрачати забагато часу спеціаліста на виконання рутинної роботи.

Тобто основними вимогами для програмного засобу для конвертації зображень у текстури є можливість одночасно працювати з великою кількістю вхідних зображень та швидкість процесу конвертації зображення.

Аналіз існуючих рішень. Аналізуючи рішення, які знаходяться у вільному доступі, їх можна умовно розділити на такі категорії: плагіни для графічних редакторів, модуль або компонента більш комплексної системи (наприклад ігрового рушія) та окремий спеціалізований програмний засіб.

Спираючись на сформовані вимоги до програмного засобу, найменш вдалим варіантом реалізації є реалізація у вигляді плагіну до графічного редактору. Основною перевагою такого підходу є уніфікація програмних засобів для спеціаліста, проте вони не мають ані можливостей для роботи з великою кількістю текстур одночасно, ані достатніх показників швидкодії, оскільки здебільшого працюють в однопоточному режимі.

У деяких випадках програмне забезпечення з конвертації зображень є складовою комплексної програмної системи. Наприклад, ігрові рушії, такі як Open 3D Engine, мають вбудовані програмні засоби для конвертації зображень у текстури [2]. Подібні програмні засоби добре адаптовані для роботи з великою кількістю зображень та мають прийнятну швидкодію, проте накладають суттєві обмеження на проєкт. Проте таке рішення не може бути інтегрованим у вже існуючий проєкт і значно зменшує кількість потенційних користувачів.

Найбільш ефективним та універсальним рішенням є реалізація програмного засобу у вигляді окремого спеціалізованого застосунку. Серед програмних рішень, що знаходяться у вільному доступі, одним з найкращих буде NVIDIA Texture Tools Exporter [3]. Даний програмний засіб надає можливість працювати з великою кількістю зображень одночасно і має непогані показники швидкодії. Проте разом з цим застосунок має суттєві вимоги до апаратного забезпечення користувача. Особливо, якщо користувач хоче скористатися можливістю виконання обчислень на графічному процесорі за допомогою бібліотеки CUDA [4].

Для отримання експериментальних результатів швидкодії розроблено набір тестових даних. Для зручності було введено наступні терміни що характеризують зображення за розміром: велике зображення, розміром 4096 на 4096 пікселі, середнє зображення розміром 1024 на 1024 пікселі та маленьке зображення розміром 256 на 256 пікселі. Тоді набір тестових даних матиме наступний вигляд:

1) 60 великих зображень;
2) 60 середніх зображень;
3) 60 маленьких зображень;
4) 20 великих зображень, 20 середніх та 20 маленьких.

Алгоритмом конвертації зображення у текстуру обрано BC1 [5]. Варто зазначити, що час конвертації залежить лише від розміру вхідного зображення, та не залежить від його вмісту.

Застосувавши програмний засіб NVIDIA Texture Tools Exporter для розробленого тестового набору даних були отримані наступні результати:
1) 60 великих зображень – 63,273 секунди;
2) 60 середніх зображень – 5,184 секунди;
3) 60 маленьких зображень – 0,541 секунди;
4) 20 великих, 20 середніх та 20 маленьких зображень – 46,687 секунди.

Отже, метою подальшого дослідження, є розробка програмного засобу, що зможе продемонструвати кращі результати на розробленому тестовому наборі даних.

Розробка архітектури програмного забезпечення. Програмне забезпечення розроблено мовою програмування C++, оскільки це одна з найшвидших сучасних мов програмування та має широкий вибір інструментів для написання багатопоточного коду у стандартній бібліотеці [6].

Для написання ефективного багатопоточного програмного засобу потрібно подбати про ефективне використання апаратних ресурсів, насамперед, під час організації управління потоками при виконанні програмного коду. Для цього був використаний шаблон програмування Thread Pool [7].

Наступним кроком необхідно провести аналіз процесу конвертації зображення у текстури, та розбити його на окремі задачі для подальшого виконання у потоках.

Весь процес можна розділити на три основні етапи (рис. 1):
1) попередня обробка зображення;
2) застосування алгоритму конвертації;
3) збереження результату у вигляді DDS текстури.

Оскільки ці етапи мають виконуватися послідовно для кожного окремого зображення, необхідно дещо модифікувати класичну реалізацію шаблону Thread Pool, для того, щоб надати можливість створювати послідовності задач для їх подальшого виконання у потоках.

Рисунок 1 – UML діаграма діяльності загального алгоритму конвертації зображення у текстуру.

Ефективність такого підходу полягає в тому, що система розподіляє навантаження між потоками для вхідних даних будь-якого розміру. Також розбиття на окремі етапи дозволяє зменшити вплив від переключення контекстів [8], що регулярно виникають, якщо певний програмний код занадто довго виконується у потоці.

Оцінка ефективності запропонованого рішення. У відповідності до описаної вище архітектури, був розроблений застосунок для конвертації зображень у текстури.

Використовуючи розроблені раніше тестові дані, отримані такі результати дослідження швидкодії обробки зображень:
1) 60 великих зображень – 33,137 секунди;
2) 60 середніх зображень – 2,296 секунди;
3) 60 маленьких зображень – 0,229 секунди;
4) 20 великих, 20 середніх та 20 маленьких зображень – 13,328 секунди.

Як можна побачити, запропоноване рішення показало набагато ефективність, в середньому скоротивши час конвертації тестового набору зображень в 2,5 рази.

Таким чином, ефективність нового програмного засобу конвертації зображень у DDS текстури доведена експериментально.

Висновки. За результатами аналізу предметної області застосування програмного забезпечення для конвертації зображень у текстури сформовані основні вимоги для даного програмного забезпечення. Виконано аналіз існуючих рішень, виділені їх переваги та недоліки. Розроблено тестовий набір даних для експериментального дослідження швидкодії програмного забезпечення.
Розроблено архітектуру програмного забезпечення для конвертації зображення у текстури. Для дослідження швидкодії розроблений застосунок у відповідності до розробленої архітектури. Ефективність запропонованого рішення доведена експериментально. В результаті час конвертації для тестового набору зображень був зменшений у 2,5 рази.

Список інформаційних джерел

1. Microsoft Learn. DDS File Layout. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/windows/win32/direct3ddds/dx-graphics-dds-pguide#dds-file-layout
2. Open 3D Engine. [Електронний ресурс]. – Режим доступу: https://www.o3de.org/
3. Nvidia Texture Tools. [Електронний ресурс]. – Режим доступу: https://developer.nvidia.com/nvidia-texture-tools-exporter
4. NVIDIA CUDA. NVIDIA Documentation Center | NVIDIA Developer. [Електронний ресурс]. – Режим доступу: https://docs.nvidia.com/cuda/
5. Microsoft Learn. Block Compression. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/windows/win32/direct3d10/d3d10-graphics-programming-guide-resources-block-compression#bc1
6. C++ Concurrency support library. cppreference.com. [Електронний ресурс]. – Режим доступу: https://en.cppreference.com/w/cpp/thread
7. C++ Thread Pool. EDUCBA. [Електронний ресурс]. – Режим доступу: https://www.educba.com/c-plus-plus-thread-pool/
8. Microsoft Learn. Windows Context Switches. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/windows/win32/procthread/context-switches

УДК 004.93

***Шульц Софія Олексіївна**, здобувач вищої освіти першого (бакалаврського) рівня, КПІ ім. Ігоря Сікорського, Україна*
***Науковий керівник: Стеценко Інна Вячеславівна** д.т.н, проф., професор кафедри інформатики та програмної інженерії, КПІ ім. Ігоря Сікорського, Україна*

МЕТОДИ МАШИННОГО НАВЧАННЯ ТА РОЗПІЗНАВАННЯ ЕМОЦІЙ З МЕДІА-ФАЙЛІВ НА ЇХ ОСНОВІ

Анотація. Проведене наукове дослідження ставить за мету аналіз методів машинного навчання для розпізнавання емоційного забарвлення медіа-файлів. Також розглянуто деякі відомі бібліотеки алгоритмів машинного навчання для мови програмування Python, що можуть бути успішно застосовані для вирішення даних задач.
КЛЮЧОВІ СЛОВА: ШТУЧНІ НЕЙРОННІ МЕРЕЖІ (ШНМ), ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ЕМОЦІЙ, ВИЯВЛЕННЯ ОБЛИЧ.

Abstract. The aim of this research is to analyse machine learning methods for emotion recognition of media files. Also, some of the well-known libraries of machine learning algorithms for Python programing language, that can be used to solve these problems, were reviewed.
KEY WORDS: ARTIFICIAL NEURAL NETWORKS (ANNS), CONVOLUTIONAL NEURAL NETWORKS (CNN), COMPUTER VISION, EMOTION RECOGNITION, FACE DETECTION.

Вступ. Збільшення об'ємів інформації, з яким ми зіштовхуємось майже що дня, вимагає винайдення нових підходів до її обробки та аналізу. І важливу роль в цьому зіграли саме обчислювальні машини, тобто комп'ютери. Тому вкрай важливо розробляти та розвивати методи обробки даних за допомоги машинного навчання. Зараз засоби штучного інтелекту (ШІ) використовують для великої кількості задач: від персонального підбору музики чи фільмів до розроблення схем різних чипів [1]. Через стрімкий розвиток ШІ було створено відкритого листа [2] що до негайного призупинення його розвитку що найменше на 6 місяців. Навіть збір та аналіз суспільної думки можна влаштувати через правильно навчені нейронні мережі (НМ), засоби комп'ютерного зору та інші підгалузі штучного інтелекту. Аналіз емоційного забарвлення медіафайлів можна використовувати низкою способів – від збору відгуку про певний товар чи послугу до підбору фільмів чи музики з певним настроєм, чи до аналізу та подальшого контролю суспільної поведінки.

Основна частина. Часто люди не розуміють, що засоби машинного навчання та штучного інтелекту це не лише славнозвісні ChatGPT чи розумні машини, котрі захоплять світ, а багато повсякденних речей. До прикладу, досвідченні гравці в комп'ютерні ігри знають, що приємна частота оновлення кадрів на секунду для гри це 60 кадрів та більше, але не всі пристрої можуть це собі дозволити, так як їм не вистачає обчислювальних потужностей для виконання певних задач відображення гри. З популяризацією ШІ з'явилась можливість «домальовувати» пропущені кадри набагато швидше, ніж прораховувати їх за всіма правилами [3]. Звісно, такі засоби на даний час мають свої недоліки: можуть з'являтися «артефакти» на зображеннях, такі як занадто розмиті чи розтягнені фрагменти, збільшення

кількості шумів чи взагалі невідповідність зображення до реальної моделі (зайві пальці чи інші деталі).

Повертаючись до теми, варто згадати таку «науку» як фізіогноміка, котра навчає як визначити емоції людини по виразу обличчя, рухам чи мовленню. Адепти цієї науки стверджують, що здатні визначити поведінкові паттерни людини за формою вух чи щелепи. Проте базове вміння зрозуміти мову тіла чи тон співрозмовника є вагомою частиною в соціалізації людини в суспільстві.

Основні людські емоції можна визначити як «щасливий», «сумний», «здивований», «спокійний». В поняття «здивований» часто додають «злий» чи «наляканий». Проте найчастіше зустрічаються в наборах даних для навчання НМ саме наступний перелік: «нейтральний» чи «спокійний» (іноді їх можуть розділяти як дві різні емоції), «щасливий», «сумний», «злий», «наляканий», «огида» та «здивований» [4]. Звісно, це не є повним описом всіх емоцій, так як всі вони не бінарне значення наявності емоції, а скоріше діапазони, які можуть поєднуватись та накладатись одне на одного.

Запис емоцій можливий за допомоги медіафайлів (фото, відео та аудіо файли, таких форматів як mp3, mp4, jpg, png, wav, mpeg та інші). Такі дані займають більше пам’яті, ніж текстові, тому їх можна віднести до задач оброблення великих обсягів даних. Хорошим рішення даної проблеми є використання методів машинного навчання. Найчастіше використовують саме згорткові НМ, так як звичайні перцептрони (одношарові НМ) здатні лише на просте лінійне розділення нелінійних множин (не можливе створення такої ШНМ для вирішення задач розділення множини на логічне «виключне або» [5]).

Згорткові НМ (з англ. Convolution Neural Network (CNN)) мають 3 типи шарів [6]. «Згортковий» шар є основою таких ШНМ та полягає в пошуку скалярного добутку двох матриць, ядра та вхідних даних. Ядро зазвичай є меншим за розмірами, але має однакову глибину. Тобто, якщо вхідні дані – зображення з трьома кольоровими каналами (rgb) розміром 400 на 400 пікселів, то ядро-матриця розмірами 5 на 5 (лише приклад) та глибиною обов’язково 3 канали (rgb). Під час прямого просування на цьому типу шарів ядро ковзає вздовж ширини та висоти вхідних даних (для зручності розглянемо їх як двовимірне зображення). Такий підхід дозволяє витрачати менше пам’яті на зберігання даних (малий розмір ядра), створювати мапи активації з повторним використанням ядра та давати еквівалентне представлення змін вхідних даних щодо результатів.

«Об’єднуючі» шари мають за мету замінити та зменшити обсяг вхідних до них даних. Тобто серед певної області пікселів (елементів матриць вхідних даних) вони за певними правилами обирають нове значення. Такими правилами можуть бути середнє значення, Евклідова метрика, зважене середнє, максимум тощо.

«Повністю з’єднані» шари – це такі, в котрих між всіма вхідними та вихідними нейронами є зв’язок. Вони використовуються для мапування отриманих результатів під припустимий формат відповіді.

Важливо вказати, що часто отримані результати згорткових шарів оброблюють нелінійними функціями для нормалізації вихідних даних. Такими функціями можуть бути сигмоїд (1), гіперболічний тангенс (2) та ReLU(3), які перетворюють вхідне число в межах [0, 1], [-1, 1], [0, MAX], де MAX – певне найбільше допустиме число [6]:

$$\sigma(k)=1/(1+e^{-k}) \quad (1)$$

$$f(k)=\tanh(k) \quad (2)$$

$$f(k)=\max(0, k) \quad (3)$$

Найпопулярнішою є функція ReLU, так як є найнадійнішою та підвищує збіжність в 6 разів, проте небезпекою її використання є можливість різкої зміни градієнту так, що відповідні ваги більше не зможуть оновлюватись, чому можна спробувати

запобігти підібравши допустиме значення швидкості навчання.

Для створення та навчання ШНМ використовують бібліотеки мови програмування Python. Наведемо найбільш відомі з них.

PyTorch – бібліотека комп'ютерного зору та обробки природної мови, є доволі складною у використанні та створена компанією Meta, підтримується багатьма хмарними платформами та операційними системами, має власну екосистему, котра спрощує її використання та розширює можливості бібліотеки.

Одна з найпопулярніших відкритих бібліотек машинного навчання – TensorFlow. Має велику кількість покрокових інструкцій для розробників різного рівня і може бути запущена на різних платформах.

Keras – API, що створене для людей, а не для машин, як стверджується в гаслі компанії. Ця бібліотека побудована на основі TensorFlow та надає можливість її спрощеного використання навіть на кластерах GPU та TPU. Використовується в багатьох організаціях типу NASA, NIH, CERN та інших.

Так як остання бібліотека є інтерфейсом, то слід порівняти попередні дві між собою. Обидві бібліотеки створені великими компаніями. Так Google, взявши за основу «Theano» створили TensorFlow, а компанія Meta (раніше FaceBook) на основі бібліотеки «Torch» створила PyTorch, не просто додавши обгортки для вже відомих функцій, а переписавши алгоритми на більш швидкі та зрозумілі. PyTorch переважно застосовується для окремих розробників і переважно наукової роботи (через те, що має велику здатність до віртуалізації, яка дозволяє запускати написані проєкти на різних пристроях), водночас TensorFlow переважно застосовується у великих та комерційний проєктах [7].

Тому при виборі фреймворку для виконання даного наукового дослідження було б зручніше обрати PyTorch, який перемагає свого конкурента в гнучкості, можливостях дебагу та швидкості тренування [7]. Проте завдяки використанню бібліотеки Keras можна урівняти положення описаних бібліотек. Тоді такі переваги TensorFlow та Keras як хороша документація, простота введення створених ШНМ в готові програмні продукти, стають вагомим аргументом в сторону цих бібліотек.

Вирішення задач розпізнавання емоцій з медіафайлів можна розділити на два типи: 1) пошук облич людей на фото та/чи відео та визначення емоційного забарвлення виразі їх обличчя; 2) аналіз тону аудіоданих для визначення емоційного забарвлення аудіо-файлів. Такі ж речі, як визначення змісту тексту аудіоданих чи виділення слів, речень в текстовий формат є задачами обробки природної мови, які в цьому дослідженні не розглядаємо.

Для аналізу аудіоданих потрібно попередньо обробити їх. Сам запис і текс на ньому не є важливими, головне – зібрати статистику, наприклад, середню гучність, середній градієнт зміни гучності тощо. Загалом для спрощення аудіоданих та їх аналізу ШНМ використовують один з видів виокремлення даних: MFCC, Spectral Centroid, Log-Mel Spectrsogram, Zero-Crossing Rate та інші.

MFCC (мел-частотні коефіцієнти кепстру) – спосіб вираховування особливостей аудіо на основі емпіричної мел-шкали (мапування частоти звуку до здатностей людського слуху) [8]. Ще один варіант обробки – це Spectral Centroid [9], який часто асоціюють з виміром яскравості звуку, тобто така обробка визначає розміщення «підвищень» звуку, часто використовується для розділення музичних творів на жанри, хоч і не має дуже великої точності. Zero-Crossing Rate [9] зрозумілий з назви і показує як часто певний сигнал перетинає значення 0. Але основним показником приналежності аудіо до якогось класу не може бути через те, що репрезентує лише невелику частину даних.

Найчастіше використовують саме MFCC [10] в доповненні з попередньою швидкою

трансформацією Фур'є (fast Fourier transform або ж FFT). Для неперіодичних сигналів, якими якраз і є живе мовлення, використовують дану трансформацію, але на невеликих проміжках часу. Така обробка дозволяє зрозуміло показати зміну гучності та/чи амплітуди відносно часу для різних частот.

Для обробки зображень з метою розпізнавання облич та емоцій на них варто використовувати засоби комп'ютерного зору [11]. Спершу потрібна попередня обробка для вхідних даних – розбиття відео на кадри та взяття лише частини з них, для підвищення продуктивності ПЗ. Якщо мережа тренувалась на чорно-білих зображеннях, то вхідні дані для роботи потрібно привести до відповідного формату. Потім необхідно визначати положення обличчя на зображенні.

Якщо обробка мережею зображень людського обличчя не сильно відрізняється від такої для будь-яких інших вхідних даних, то виділення обличчя з фото є окремою категорією задач в галузі комп'ютерного зору. Способи вирішення цієї задачі ділять на чотири групи [12]: засновані на знаннях, на вигляді, на основі особливостей та такі, що підпадають під шаблон. Методи, засновані на знаннях, мають під собою певний набір правил, наприклад: обличчя має мати ніс, рот та два ока на певній відстані і тд. Тобто правила, які просто описати людині, але важко записати програмно. Методи на основі особливостей шукають певні риси обличчя, тобто класифікують участки зображення як ті, де є ці риси і де їх немає. Деякі такі системи правильно визначають до 94% випадків. Методи підпадання під шаблони мають генералізовані шаблони вигляду облич під різними кутами і просто шукають співпадіння на отриманих фото. І найбільш точні методи засновані на вигляді. Вони використовують попередньо навчені моделі для пошуку облич на зображенні, здатні зчитувати різні особливості зображення, використовуючи такі алгоритми та методи, як визначення меж, нейронні мережі, наївний Баєсів класифікатор та інші. Вибір відповідного алгоритму залежить від потрібного співвідношення швидкості роботи до якості чи від певних фізичних обмежень обчислювальних машин.

Висновки. Отже, в сьогоденних реаліях маємо, що використання штучних нейронних мереж можна зустріти в багатьох галузях нашого буденного життя. Як для розробників, варто звернути свою увагу на бібліотеки PyTorch та TensorFlow (з використанням API Keras). Кожна з них має свої переваги та недоліки, але для ознайомлення та наукових робіт, особливо в галузі комп'ютерного зору, буде зручнішим у використанні перший варіант, коли ж більш корпоративні рішення краще створюються на основі другої бібліотеки. Одношарові найпростіші мережі не підійдуть для такого розпізнавання, тому хорошим рішенням буде використати згорткові нейронні мережі. Попередня обробка вхідних даних має включати в себе відповідне виокремлення особливостей (пікових значень, обробки аудіо алгоритмами MFCC чи визначенням положення обличчя, що найкраще робити методами комп'ютерного зору, котрі засновані на вигляді). Також зрозуміло, що розпізнавання емоційного забарвлення медіа-файлів, особливо з режимі реального часу, є нагальною потребою, до вирішення якої докладають сил і великі корпорації, надаючи можливість гнучкої розробки програмних продуктів для такого типу розпізнавання.

Список інформаційних джерел

1.Anna Goldie. Chip Design with Deep Reinforcement Learning [Електронний ресурс] / Anna Goldie // Google Research. – 2020. – Режим доступу до ресурсу: https://ai.googleblog.com/2020/04/chip-design-with-deep-reinforcement.html.

2. Pause Giant AI Experiments: An Open Letter [Електронний ресурс] // Future of Life. – 2023. – Режим доступу до ресурсу: https://futureoflife.org/open-letter/pause-giant-ai-experiments/.
3. Ted Xiao. Frame Rate Upscaling with Deep Neural Networks / Ted Xiao, Raul Puri, Gautham Kesineni., 2015. – (Machine Learning at Berkeley).
4. A Model for Basic Emotions Using Observations of Behavior in Drosophila / Simeng Gu, Fushun Wang, Nitesh P. Patel та ін.]., 2019. – (frontiers in psychology).
5. Rossella Cancelliere. Multi layer feed-forward NN XOR problem [Електронний ресурс] / Rossella Cancelliere // University of Turin – Режим доступу до ресурсу: http://www.di.unito.it/~cancelli/retineu11_12/FNN.pdf.
6. Pranshu Sharma. Basic Introduction to Convolutional Neural Network in Deep Learning [Електронний ресурс] / Pranshu Sharma // Analytics Vidhya. – 2022. – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-convolutional-neural-network-in-deep-learning/.
7. John Terra. Keras vs Tensorflow vs Pytorch: Key Differences Among Deep Learning [Електронний ресурс] / John Terra // Simpl I learn. – 2023. – Режим доступу до ресурсу: https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article.
8. Uday Kiran. MFCC Technique for Speech Recognition [Електронний ресурс] / Uday Kiran // Analytics Vidhya. – 2021. – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/.
9. Unjung Nam. Special Area Exam Part II / Unjung Nam., 2001. – 10 с. – Режим доступу до ресурсу: https://ccrma.stanford.edu/~unjung/AIR/areaExam.pdf
10. Leland Roberts. Understanding the Mel Spectrogram [Електронний ресурс] / Leland Roberts // Analytics Vidhya. – 2020. – Режим доступу до ресурсу: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53.
11. Mingjie Wang. Facial expression recognition based on CNN / Mingjie Wang. // Journal of Physics: Conference Series. – 2020. – С. 6.
12. Divyansh Dwivedi. Face Detection For Beginners [Електронний ресурс] / Divyansh Dwivedi // Towards Data Science Towards Data Science Your home for data science. A Medium publication sharing concepts, ideas and codes. Towards Data Science Followed by 661030 people Follow. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9.

УДК 004.416

Похиленко Олександр Андрійович, *здобувач вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник: Баклан Ігор Всеволодович, *кандидат технічних наук,*
доцент, доцент кафедри інформатики та програмної інженерії
КПІ ім. Ігоря Сікорського, Україна

МЕТОД ТА ЗАСІБ СУПРОВОДЖЕННЯ ЕВОЛЮЦІОНУЮЧИХ ГІБРИДНИХ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ

Анотація. Еволюціонуючі гібридні інтелектуальні системи (ЕГІС) є високоадаптивними системами, що здатні оновлювати значення власних параметрів та структуру на основі потоків даних. Через здатність адаптації до мінливого середовища, використання ЕГІС стає все більш популярним у багатьох різних сферах. При цьому ця особливість також робить супроводження ЕГІС складною задачею. Для вирішення цієї проблеми в цій роботі пропонується метод супроводження ЕГІС на основі виявлення аномалій. Результатами роботи є Python пакет та модель виявлення аномалій, яка може працювати без попереднього навчання, але також може підвищувати власну точність при навчанні на тренувальних даних.
КЛЮЧОВІ СЛОВА: еволюціонуюча гібридна інтелектуальна система, прогностичне супроводження, виявлення аномалій, часові ряди.

Abstract. Evolving hybrid intelligent systems (EHIS) are highly adaptive systems capable of updating the values of their own parameters and structure based on data streams. Due to their ability to adapt to a changing environment, the usage of EHIS is becoming increasingly popular in many different fields. At the same time, this feature also makes EHIS maintenance a difficult task. To solve this problem, this paper proposes a method based on anomaly detection for EHIS maintenance. The results of the work are a Python package and an anomaly detection model that can work without prior learning but can also increase its own accuracy when learning on training data.
KEY WORDS: evolving hybrid intelligent system, predictive maintenance, anomaly detection, time series.

Вступ. ЕГІС є системами, які зазвичай складаються з багатьох компонентів та моделей, значення параметрів яких можуть змінюватися із часом. Ці системи мають високу адаптивність і можуть вирішувати складні завдання в умовах обмеженої та нечіткої інформації, однак через перераховані особливості їхнє супроводження є проблематичним. Відповідно до [1], при супроводженні таких складних інтелектуальних систем рекомендується використання прогностичного супроводження на основі методів виявлення аномалій, застосування яких в контексті супроводження розглядається в [2, с. 6]. Найбільш цікавим методом, що може застосовуватися для супроводження ЕГІС, є метод на основі однокласових SVM, описаний в [3]. Особливістю цього методу є можливість не тільки виявлення, але й класифікації виявлених аномалій. При цьому запропонована в статті модель для виявлення аномалій не потребує попереднього навчання. Така особливість робить цей метод дуже корисним при використанні для супроводження нових ЕГІС, оскільки в такому випадку відсутні дані, які можуть використовуватися для навчання моделі. Таким чином, в даній роботі було вирішено узяти цей метод супроводження та удосконалити його.

Основна частина. Опис роботи було вирішено розділити на 3 головні частини. Першою частиною є удосконалення методу супроводження з метою адаптації його використання під ЕГІС. Ця частина включає додавання нових етапів до початкового

методу супроводження, розглянутому в [3]. Другою частиною є розробка моделі виявлення аномалій, що відповідає удосконаленому методу. Ця частина також включає проведення експериментів з метою порівняння ефективності удосконаленої моделі та початкової. Третьою частиною є розробка засобу супроводження. В цій частині описується вигляд засобу та рекомендації щодо його використання.

Метод супроводження ЕГІС. Початковий метод супроводження, запропонований в [3], мав 2 основних етапи: етап виявлення аномалій та етап класифікації аномалій. При цьому етап класифікації аномалій був необов'язковим і використовувався лише у разі налаштування набору класів людиною. Оскільки одноклассовий SVM належить до методів навчання без вчителя, то модель виявлення аномалій не потребувала початкового навчання на наборі позначених тренувальних даних. Однак класифікатор, у разі встановлених людиною класів, потребував навчання. Таким чином, крім 2 основних етапів, в початковому методі можна виокремити ще додатковий етап перенавчання моделі класифікатора. Неможливість перенавчити модель виявлення аномалій є слабким місцем цього методу, оскільки у разі низької точності виявлення аномалій модель буде майже неможливо покращити, окрім як змінюючи її гіперпараметри.

Отже, з метою виправлення цього недоліку було вирішено додати до удосконаленого методу етап перенавчання обидвох моделей: і моделі виявлення аномалій, і моделі класифікації. При цьому треба було зберегти можливість запуску моделі без обов'язкового початкового навчання. Такі вимоги призвели до необхідності зміни моделі виявлення аномалій, про що розписано в наступній частині.

При розробці удосконаленого методу супроводження ЕГІС було також розглянуто засіб Driftage з [4]. Метод супроводження, що використовувався в цьому засобі, включав етап збереження аномальних даних та сповіщення про аномалії чи виконання заданих дій. Враховуючи, що удосконалений метод супроводження ЕГІС має етап перенавчання моделей, то додавання до методу етапу збереження даних є доцільним, оскільки це дозволяє формувати набір тренувальних даних. Крім цього, відправка сповіщення про аномалії операторам ЕГІС також є важливим, оскільки це може пришвидшити виправлення проблем, що були причиною або наслідком цих аномалій. Автоматичне виконання заданих дій, яке згадувалося на цьому етапі, можна розглядати як різновид сповіщень, тільки в даному випадку отримувачем сповіщення є не людина, а автоматизована система, яка запускає виконання певних дій.

Ще одним етапом, який було вирішено додати до удосконаленого методу супроводження, є отримання зворотного зв'язку, який розглядався в [5]. Цей етап дозволяє виконувати позначення виявлених аномалій як істинні та хибно позитивні. В результаті позначений набір даних може бути застосовано на етапі перенавчання моделей з метою підвищення їхньої точності.

Таким чином, удосконалений метод супроводження ЕГІС складається з наступних етапів:

- виявлення аномалій;
- класифікації виявлених аномалій;
- збереження аномалій та сповіщення;
- отримання зворотного зв'язку;
- перенавчання моделей.

Модель виявлення аномалій. Як вже згадувалося в попередній частині, удосконалення методу супроводження призвело до необхідності у зміні моделі виявлення аномалій. Було вирішено застосувати ансамблевий підхід, який полягає в застосуванні декількох детекторів аномалій D замість одного. Результати, отримані від кожного детектору, нормалізуються за допомогою калібраторів C, після чого виконується множення нормалізованих значень на ваги W. Далі обчислюється

результуюче значення *r* як сума значень, отриманих на попередньому етапі, та значення коефіцієнту зсуву *b*. Останнім кроком є застосування логістичної функції активації до значення *r* з подальшим округленням до найближчого цілого числа. В такому випадку значення 1 буде відповідати виявленню аномалії, а 0 – відсутності.

При ініціалізації ваг *W* та коефіцієнту зсуву *b* початковими значеннями ця модель може виявляти аномалії за рахунок ансамблю детекторів *D*, що засновані на використанні методів навчання без вчителя. Однак при цьому значення *W* та *b* можуть бути змінені при навчанні на наборі тренувальних даних за рахунок використання моделі лінійної регресії. За рахунок цього точність отриманої моделі може підвищуватися в результаті навчання.

Приклад отриманої удосконаленої моделі для набору двовимірних вхідних значень *X* зображений на рис. 1.

Для того, щоб перевірити розроблену модель, було вирішено провести експеримент на наборі даних Yahoo для виявлення аномалій в часових рядах [6]. В результаті цього експерименту виявилося, що розроблена удосконалена модель має таку ж повноту (recall), як і початкова модель, однак після навчання її влучність (precision) в 2.16 разів перевищує відповідне значення початкової моделі.

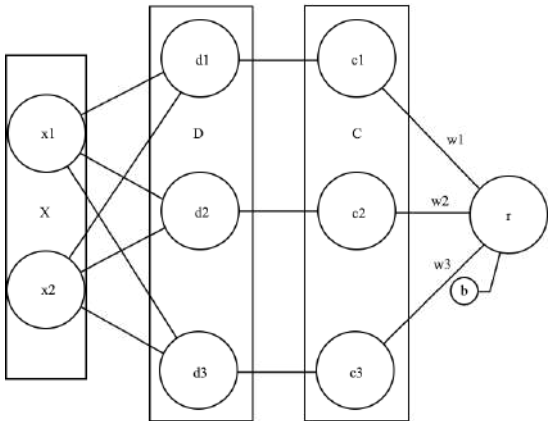


Рис. 1. Модель виявлення аномалій

Засіб супроводження ЕГІС. Засіб супроводження ЕГІС було вирішено розробити у вигляді 2 модулів: бібліотеки (пакету Python) та веб-застосунку (Django застосунку). При розробці засобу використовувалася бібліотека StreamAD для виявлення аномалій в потоках даних у вигляді часових рядів [7]. Веб-застосунок було створено з метою надання зручного інтерфейсу для керування моделями виявлення аномалій. Цей застосунок може виявитися корисним у разі супроводження складної ЕГІС, що має декілька незалежних потоків даних, які слід перевіряти на наявність аномалій окремо. В той же час бібліотека може бути використана окремо від засобу, оскільки була опублікована в репозиторії пакетів PyPI. Це дозволяє встановити бібліотеку і впровадити її у вже існуючу ЕГІС чи засіб супроводження ЕГІС.

Висновки. В результаті проведеної роботи було запропоновано метод супроводження ЕГІС, який був створений шляхом удосконалення методу прогностичного супроводження на основі виявлення та класифікації аномалій. Також було запропоновано модель виявлення аномалій, що використовує ансамбль детекторів, заснованих на використанні методів навчання без вчителя. Результуюча модель може працювати без попереднього навчання, однак дозволяє підвищувати власну точність виявлення аномалій за рахунок застосування моделі лінійної регресії. Експериментальні дослідження продемонстрували, що отримана модель має таку ж повноту, як і початкова модель з [3], а після навчання влучність моделі в 2.16 разів перевищує влучність початкової моделі. Це свідчить про значне зменшення хибно позитивних спрацювань на етапі виявлення аномалій. Крім цього було розроблено засіб супроводження ЕГІС, що реалізує розглянутий метод та модель.

Список інформаційних джерел

1. Lima A. L. da C. D., Aranha V. M., Carvalho C. J. de L., Nascimento E. G. S. Smart predictive maintenance for high-performance computing systems: a literature review. *The Journal of Supercomputing*. 2021. Vol. 77. No. 11. P. 13494-13513.
2. Sharma J., Mittal M. L., Soni G. Condition-based maintenance using machine learning and role of interpretability: a review. *International Journal of System Assurance Engineering and Management*. 2022. P. 1-16.
3. Morselli F., Bedogni L., Mirani U., Fantoni M., Galasso S. Anomaly Detection and Classification in Predictive Maintenance Tasks with Zero Initial Training. *IoT*. 2021. Vol. 2. No. 4. P. 590-609.
4. Vieira D. M., Fernandes C., Lucena C., Lifschitz S. Driftage: a multi-agent system framework for concept drift detection. *GigaScience*. 2021. Vol. 10. No. 6.
5. Holzinger A. The Next Frontier: AI We Can Really Trust. *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. 2021. P. 427-440.
6. Laptev N., Amizadeh S., Billawala Y. A Benchmark Dataset for Time Series Anomaly Detection [Електронний ресурс] – Режим доступу: https://yahooresearch.tumblr.com/post/114590420346/a-benchmark-dataset-for-time-series-anomal.
7. StreamAD documentation [Електронний ресурс] – Режим доступу: https://streamad.readthedocs.io/

УДК 004.91

Бондаренко Дмитро Сергійович, *здобувач вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник: Ліхоузова Тетяна Анатоліївна, *кандидат технічних наук,*
доцент, доцент кафедри інформатики та програмної інженерії
КПІ ім. Ігоря Сікорського, Україна

ОГЛЯД ПРОГРАМНИХ ЗАСОБІВ ОБЛІКУ КРИПТОВАЛЮТНИХ ОПЕРАЦІЙ

Анотація. Одним із завдань, яке стає перед криптоінвесторами, є ведення обліку власного портфелю, щоб мати всю інформацію про власні вклади та їх статус (в найпростішому випадку це ціни). Щоб спростити процес, використовують різні програмні засоби, які дозволяють користувачам відстежувати свої інвестиції, стежити за змінами цін на криптовалюту та переглядати ефективність свого портфеля. Вони також надають користувачам уявлення та аналітику про їхні інвестиції, що полегшує прийняття обґрунтованих інвестиційних рішень. Це важливий інструмент для інвесторів, необхідний для ефективного управління інвестиціями, тому варто обирати його зважено.
КЛЮЧОВІ СЛОВА: засоби обліку, криптовалюта, інвестиційний портфель.

Abstract. One of the tasks facing crypto-investors is keeping records of their own portfolio to have all the information about their contributions and their status (in the simplest case, it prices). To simplify the process, various software tools are used that allow users to track their investments, monitor cryptocurrency price changes, and view the performance of their portfolios. They also provide users with insights and analytics about their investments, making it easier to make informed investment decisions. It is an important tool for investors, and necessary for effective investment management, so it should be chosen carefully.
KEY WORDS: accounting tools, cryptocurrency, investment portfolio.

Вступ. В реаліях сьогодення громадяни все частіше перестають довіряти звичайним фіатним валютам як засобу збереження власних статків. Ріст інфляції з кожним роком стає все швидше, що спонукає шукати альтернативні способи збереження або примноження власних капіталів. Хтось інвестує у нерухомість або вкладає гроші у дорогоцінні метали, а є ті, хто обирає новий та ще не такий всесвітньовідомий і популярний варіант – купівлю криптовалют.

Основна частина. Купувати криптовалюти людей спонукає не лише задача збереження статку. Так, наприклад, криптовалюти є децентралізованими. Це означає, що вони не контролюються урядом або фінансовою установою. Криптовалюти також мають нижчу комісію за транзакції та пропонують практично миттєвий час розрахунків. [1] Нарешті, зростання визнання та застосування криптовалют як законної форми платежу в деяких країнах викликало ажіотаж серед населення, а це у свою чергу спричиняє ріст цін на них.

Криптовалют існує дуже багато і кожна із них має свою ціну. Це дозволяє з легкістю диверсифікувати власні вклади. Коли одна із валют може впасти у ціні, то інша у цей час може, навпаки, вирости.

При наявності куплених декількох криптовалют у власному володінні, виникає проблема зручності їх обліку. Не дуже зручно тримати у пам’яті або десь записувати інформацію про те, на якій біржі була куплена та чи інша валюта. Також треба мати актуальні знання про статус та ціну тої чи іншої валюти. Тому проблема обліку криптовалют та операцій, які користувач проводив із ними, є актуальною і потребує простого та зручного вирішення.

На даний момент на ринку вже наявні веб-застосунки із частиною такого функціоналу, але кожен із них має свої переваги та недоліки.

Як таких технічних рішень, які б надавали саме функції обліку криптовалют, немає. Є допоміжні технології, такі як криптовалютні API, які надають потрібну інформацію про валюти (ціни на звичайні валюти, ціни на криптовалюти, загальний список). Найпоширеніші з них CoinMarketCap API, CryptoCompare API та CoinGecko API.

CoinMarketCap API [2]. Переваги:

- вичерпна та зрозуміла документація;
- велика база даних із криптовалютами;
- багато кінцевих точок для запитів.

Недоліки:

- безкоштовна версія обмежена 10,000 запитами на місяць.

CryptoCompare API [3]. Переваги:

- кінцеві точки мають у собі багато необов'язкових параметрів, що робить їх використання гнучким.

Недоліки:

- безкоштовна версія обмежена 100,000 запитами на місяць;
- гнучка, але не дуже зрозуміла документація.

CoinGecko API [4]. Переваги:

- вичерпна та зрозуміла документація;
- велика база даних із криптовалютами;
- не потребує реєстрації та використання API ключа.

Недоліки:

- безкоштовна версія обмежена 10-30 запитами на хвилину.

Із наведених API для розробки засобу для обліку варто використати останній із них, оскільки він не потребує використання API ключа та не обмежений по кількості запитів на місяць.

Після аналізу відомих веб-додатків було обрано 3 приклади: CoinGecko Portfolio, CoinMarketCap Portfolio Tracker, CoinStats Portfolio.

CoinGecko [5]— це агрегатор даних про криптовалюти та аналітична платформа, яка надає дані та статистику в реальному часі про різні криптовалюти, біржі та токени. Запущена в 2014 році, вона стала однією із широко використовуваних платформ для відстеження та аналізу ринку криптовалют. CoinGecko надає широкий спектр даних, включаючи ринкову капіталізацію, обсяги торгів та інформацію про ціни. Окрім можливостей відстеження даних, CoinGecko також пропонує різноманітні інструменти та ресурси, які допомагають користувачам приймати обґрунтовані рішення щодо своїх інвестицій. Одними із таких інструментів є API та сервіс для обліку криптовалютних заощаджень - CoinGecko Portfolio.

Із переваг:

- абсолютно безкоштовна
- підтримка величезної кількості криптовалют.

Із недоліків:

- не зручний та обмежений функціонал (треба спочатку додати криптовалюту, а потім у неї додавати транзакції; потім цю криптовалюту неможливо видалити із портфоліо);
- часті «баги», додаток не вірно себе поводить якщо обрати базову валюту відмінну від долара США.

CoinMarketCap [6] — це веб-сайт, який надає дані та інформацію про різні криптовалюти в режимі реального часу, включаючи їх ринкову капіталізацію, ціну, обсяг тощо. Це одна з найбільш широко використовуваних платформ у індустрії криптовалют і вважається надійним джерелом даних і статистики про криптовалюту. Платформа була заснована в 2013 році і з тих пір стала одним з провідних джерел для відстеження ефективності криптовалют. [Додати посилання] Окрім ринкових даних, CoinMarketCap також надає користувачам новини та освітні ресурси про індустрію криптовалют. Завдяки зручному інтерфейсу та вичерпним даним CoinMarketCap став цінним ресурсом для окремих осіб та установ, зацікавлених у тому, щоб бути в курсі останніх подій у світі криптовалют. Так само, як і CoinGecko, даний ресурс надає власне API та інструмент для обліку власних вкладів CoinMarketCap Portfolio Tracker.

Із переваг:

- абсолютно безкоштовна;
- дуже зручний та зрозумілий інтерфейс.

Із недоліків:

- немає підтримки деяких непопулярних криптовалют; на самому сервісі вони є, але неможливо створити із ними транзакцію у трекері.

CoinStats [7] — це інструмент для управління портфелем криптовалют та аналізу ринку. Він пропонує користувачам централізовану платформу для відстеження своїх криптовалютних активів, моніторингу цін і ринкових даних у режимі реального часу та отримання останніх новин і подій у криптовалютному світі.

Із переваг:

- підтримка величезної кількості криптовалют.

Із недоліків:

безкоштовна версія обмежена 1000 транзакціями та 10 портфелями.

Висновки. Як показав базовий аналіз вже наявних веб-засосунків, всі вони мають у собі якісь нюанси, по типу неприємних «багів», не дуже приємного користувацького інтерфейсу, відсутності підтримки непопулярних валют або взагалі є небезкоштовними. Виходячи із цього, постає проблема розробки такої платформи, яка б дозволила поєднати в собі зручність, доступність та мати функціонал для зручного моніторингу та обліку криптовалют.

Синхронізація із загальним ринком криптовалют має бути автоматичною, тобто нові криптовалюти та інформація про них мають додаватися у додаток автоматично. Це позбавить потреби підтримки застосунку зі сторони адміністрації та допоможе заощадити кошти на розробці та підтримці. Основний користувач системи – криптоінвестор. У його основні задачі входить створення власного портфоліо та додавання до них транзакцій виду «покупка», «продаж», «трансфер».

Список інформаційних джерел

1. Роль криптовалюти у цифровій економіці / С.В. Мерінова, Л.П. Половенко // Вісник ХДУ Серія Економічні науки. – 2021. – № 42. – С. 80-87.
2. CoinMarketCap API Електронний ресурс. – URL: https://coinmarketcap.com/api/documentation/v1/
3. CryptoCompare API Електронний ресурс. – URL: https://min-api.cryptocompare.com/documentation
4. CoinGecko API Електронний ресурс. – URL: https://www.coingecko.com/en/api/documentation
5. CoinGecko Електронний ресурс. – URL: https://www.coingecko.com/en/portfolios/
6. CoinMarketCap Електронний ресурс. – URL: https://coinmarketcap.com/portfolio-tracker/
CoinStats Електронний ресурс. – URL: https://coinstats.app/portfolio/

УДК 004.932.2

Смілянець Федір Андрійович, *аспірант,*
КПІ ім. Ігоря Сікорського, Україна
Фіногенов Олексій Дмитрович, *кандидат технічних наук, доцент*
КПІ ім. Ігоря Сікорського, Україна

МУЛЬТИКЛАСОВА КЛАСИФІКАЦІЯ ЛЕГЕНЕВИХ ЗАХВОРЮВАНЬ ЗА ДОПОМОГОЮ ЗНІМКІВ КОМП'ЮТЕРНОЇ ТОМОГРАФІЇ

Анотація. Існуючу архітектуру нейронної мережі для бінарної класифікації зображень КТ між коронавірусною пневмонією та здоровими легенями було модифіковано та адаптовано для розрізнення між трьома класами: COVID-19, здорові легені, позагоспітальна пневмонія. Дану нейронну мережу було натреновано за допомогою значно розширеного набору даних, та досягнуто точності у 95%, що наближається до зразкових в індустрії результатів.
КЛЮЧОВІ СЛОВА: мультикласова класифікація, згорткові нейронні мережі, аналіз знімків комп'ютерної томографії.

Abstract. The existing neural network architecture for binary classification of CT images between coronavirus pneumonia and healthy lungs has been modified and adapted to distinguish between three classes: COVID-19, healthy lungs, and community-acquired pneumonia. This neural network was trained using a significantly expanded dataset, achieving an accuracy of 95%, which approaches industry-leading results.
KEY WORDS: multiclass classification, convolutional neural networks, computed tomography scan analysis

Вступ. Розповсюдження в 2019-2023 роках пандемії нового коронавірусу COVID-19 призвело до різкого зростання навантаження на лікарів та медичні установи. На початок травня 2023 року більше 750 мільйонів випадків хвороби було зареєстровано, та майже 7 мільйонів мали летальний наслідок.
Попри оголошення ВООЗ про закінчення глобальної пандемії COVID-19 5 травня 2023 року, легеневі захворювання все ще мають та матимуть у майбутньому потенціал до спричинення епідемій та пандемій з відповідними наслідками для систем охорони здоров'я, відповідно актуальним є подальше дослідження систем, які дозволяють частково розвантажити та оптимізувати роботу закладів охорони здоров'я.
Знімки КТ є одним з основних методів діагностики захворювань легень, оскільки дає тривимірне та детальне зображення, процес зняття якого може бути підлаштований для процесу діагностики конкретного захворювання, а також дає легку можливість для побудови довільних зрізів простору, відзнятого знімком КТ.

Основна частина. У досліджені було об'єднано два набори даних: COVID-CTset [1](набір А) та COVID-CT-MD (набір Б). Набір А складається з зображень, що належать 95 пацієнтам з діагностованим COVID-19, та 282 здоровим людям. Оскільки DICOM-зображення містять особисті дані пацієнтів, та цей формат є відносно незручним для роботи – дані постачається у вигляді 16-бітних чорно-білих TIFF-зображень, що гарантує збереження повної глибини кольору наявної в DICOM-зображеннях.
Однак, набір А є відносно невеликим джерелом даних, та не містить знімків людей з звичайними негоспітальними пневмоніями,

тому до джерельних даних до навчання було додано датасет набір Б.

Набір Б містить зображення, що належать 169 людям з діагностованим COVID-19, 76 здоровим людям та 60 людям з діагностованою негоспітальною пневмонію, та постачається у форматі DICOM. Відтак, зображення було приведено до 16-бітного TIFF за допомогою модифікованого коду бібліотеки dcm2hdr [3], та додано до даних набору А.

Знімок КТ є набором послідовних знімків перерізу тіла людини. У випадку діагностики захворювань легень, виконується КТ грудної клітини, однак не всі відзняті знімки включають в себе внутрішню структуру легень. Тому, за алгоритмом, запропонованим у [1], було виконано фільтрацію знімків та виокремлено знімки, на яких істотну частину зображення займають саме легені. На (рис. 1) проілюстровано відмінність між зображеннями з різних частин тривимірного КТ-знімку.

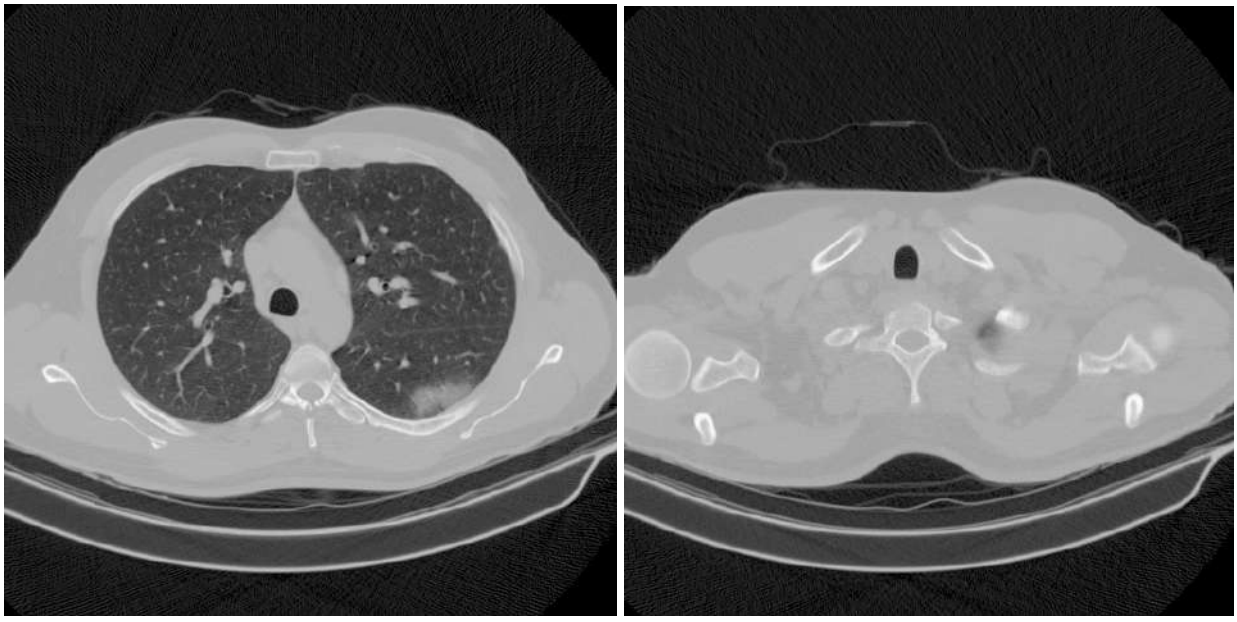


Рисунок 1. Порівняння знімків КТ грудної клітини на різних зрізах відсканованого простору

У дослідженні використовується модифікована версія мережі, запропонованої у [1].

Однак, оригінальне дослідження обмежувалось двома классами – здоровий або COVID-19, що суттєво обмежує застосування у реальному світі.

Відтак, в цьому дослідженні існуючу архітектуру нейронної мережі було розширено для виконання мультикласової класифікації, та натреновано за допомогою значно більшого набору даних. Доповнена нейронна мережа побудована з метою відрізняти між COVID-19, позагоспітальною пневмонією та здоровими легенями.

За основу було взято стандартну архітектуру ResNet50V2. Однак, оскільки прояви як типових, так і спричинених COVID-19 пневмоній мають різні особливості та масштаби, було застосовано метод Feature Pyramid Network – починаючи з третього шару мережі результат роботи кожного згорткового шару мережі передано до останніх, повнозв’язних, шарів мережі що приймають кінцеве рішення про наявність чи відсутність у зображенні особливостей, пов’язаних з пневмонією, спричиненою COVID-19 або типовою пневмонією.

Для тренування нейронної мережі було обрано стартові ваги аналогічної нейронної мережі, натренованої на датасеті ImageNet. Дані, що є відібраними з датасетів зображеннями що містять відкриті легені, було розділено на тренувальний, валідаційний та оціночний набори даних – 20701 тренувальних, 6900 валідаційних та

6900 оціночних зображень. Тренування було проведено протягом 20 епох з відбором найкращих мереж за допомогою валідаційного набору даних.

За результатом було обрано модель станом на кінець 13 епохи навчання та точністю на валідаційних даних у 95.086%.

Для обчислення істинних параметрів поведінки моделі через обрану модель було пропущено оціночний набір даних, та підтверджено точність у 95%.

Також, було обчислено матрицю невідповідності для усіх класів в дослідженні.

COVID-19	Прогнозований клас		
Справжній клас		Позитивний	Негативний
	Позитивний	2707	83
	Негативний	231	3879

Таблиця 1. Матриця невідповідності для класу «пневмонія, спричинена COVID-19»

Здорові легені	Прогнозований клас		
Справжній клас		Позитивний	Негативний
	Позитивний	3118	204
	Негативний	93	3485

Таблиця 2. Матриця невідповідності для класу «Здорові легені»

Позагоспітальна пневмонія	Прогнозований клас		
Справжній клас		Позитивний	Негативний
	Позитивний	3118	204
	Негативний	93	3485

Таблиця 3. Матриця невідповідності для класу «позагоспітальна пневмонія»

Висновки. Отримана нейронна мережа для розпізнавання COVID-19 на зображеннях КТ демонструє високу точність класифікації, наближаючись до зразкових результатів в індустрії, та може стати корисним інструментом для швидкої та ефективної діагностики COVID-19, та основою для нейронних мереж, що розпізнаватимуть інші захворювання.

Список інформаційних джерел

1. A fully automated deep learning-based network for detecting COVID-19 from a new and large lung CT scan dataset. / Rahimzadeh, M., Attar, A., & Sakhaei, S. // Biomedical Signal Processing and Control – 2021 – С. 102588 – https://doi.org/10.1016/j.bspc.2021.102588.

2. COVID-CT-MD, COVID-19 computed tomography scan dataset applicable in machine learning and deep learning / Afshar, P., Heidarian, S., Enshaei, N., Naderkhani, F., Rafiee, M., Oikonomou, A., Fard, F., Samimi, K., Plataniotis, K., & Mohammadi, A. // Scientific Data – 2021 – Т. 8(1) – С. 121 – (https://doi.org/10.1038/s41597-021-00900-3).

3. DCM2HDR: DICOM to HDR image conversion. / David Völgyes. // – 2018. – https://doi.org/10.5281/zenodo.1246724

УДК 004.414

***Гнатченко Дмитро Дмитрович**, старший викладач*
кафедри інженерії програмного забезпечення та кібербезпеки
Державний торговельно-економічний університет, Україна
***Корчага Тетяна Анатоліївна**, здобувач вищої освіти*
Державний торговельно-економічний університет, Україна

ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ JAVA-ДОДАТКІВ ЗА ДОПОМОГОЮ РЕАКТИВНОГО ПРОГРАМУВАННЯ

INCREASING THE PRODUCTIVITY OF JAVA APPLICATIONS USING REACTIVE PROGRAMMING

Анотація. Швидке зростання кількості користувачів та посилення вимог до якості застосунків змушує програмістів ширше використовувати реактивне програмування для реалізації високопродуктивних та стабільних застосунків, які адаптовані до сучасних вимог та стандартів. В роботі розглянуто плюси та мінуси такого програмування, а також висвітлено найпоширеніші методи та техніки цього напрямку.
КЛЮЧОВІ СЛОВА: реактивне програмування, асинхронне програмування, неблокуючий ввід-вивід, зворотний тиск, реактивні потоки, кешування, масштабована програма, фреймворки.

Abstract. The rapid growth of the number of users and increased requirements for the quality of applications forces programmers to use reactive programming more widely to implement high-performance and stable applications that are adapted to modern requirements and standards. The work considers the pros and cons of such programming, as well as highlights the most common methods and techniques in this area.
KEY WORDS: reactive programming, asynchronous programming, non-blocking I/O, back pressure, reactive streams, caching, scalable program, frameworks.

Вступ. Java – це потужна мова програмування, яка використовується для розробки великої кількості програм. Однак із зростанням складності сучасних застосунків і потребою в оперативному реагуванні в режимі реального часу, розробники почали звертатися до реактивного програмування. Покращення продуктивності Java-додатків за його допомогою вимагає використання кількох методів, таких як асинхронне програмування, неблокуючий ввід-вивід, зворотний тиск, реактивні потоки, кешування та реактивні бази даних. Використовуючи їх, розробники можуть покращити швидкість, стійкість і масштабованість своїх програм.

Основна частина. Реактивне програмування – це парадигма програмування, яка дозволяє розробникам писати більш стійкі та масштабовані програми. Далі розглянемо, як можна покращити продуктивність програм Java за допомогою реактивного програмування. Для цього детальніше зупинимося на кожному з методів.

Асинхронне програмування. Дана техніка дозволяє розробникам звільнити потоки та запобігти блокуванню, що може покращити роботу програми. Java забезпечує підтримку асинхронного програмування за

допомогою CompletableFuture, яка є частиною стандартної бібліотеки Java 8. Однак реактивні фреймворки програмування, такі як RxJava, Reactor і Akka, забезпечують більш розширену підтримку асинхронного програмування [1].

Неблокуючий ввід-вивід. Цей метод дозволяє програмі продовжувати обробку запитів, очікуючи завершення операцій введення-виведення. Це може підвищити продуктивність програми, запобігаючи блокуванню програми під час очікування та завершення операцій. Java забезпечує підтримку неблокуючого вводу-виводу за допомогою фреймворка Netty [2].

Зворотний тиск. Це прийом, який дозволяє споживачеві контролювати швидкість, з якою він отримує дані від виробника. Використовуючи зворотний тиск, розробники можуть запобігти перевантаженню своєї програми даними, що може підвищити її стійкість [3].

Реактивні потоки. Це специфікація асинхронної обробки потоків із неблокуючим зворотним тиском [4]. Використовуючи реактивні потоки, розробники можуть писати код, який можна складати та використовувати багаторазово.

Кешування. Метод, який може підвищити продуктивність програми шляхом зберігання даних, до яких часто звертаються, у пам’яті. Використовуючи реактивне програмування, розробники можуть кешувати дані асинхронно та уникати блокування програми. Java забезпечує підтримку кешування за допомогою фреймворків Ehcache і Hazelcast [5].

Реактивні бази даних. Реактивне програмування дозволяє розробникам використовувати реактивні бази даних, такі як MongoDB і Cassandra [6]. Ці БД призначені для обробки асинхронного неблокуючого введення-виведення. Використовуючи їх, розробники можуть скористатися перевагами асинхронних можливостей і підвищити продуктивність своїх програм.

Інша важлива перевага – здатність обробляти паралельність і паралелізм. Паралельність означає здатність програми виконувати кілька завдань одночасно, тоді як паралелізм означає здатність програми використовувати кілька процесорів для виконання завдання [7]. Також реактивне програмування забезпечує більш декларативну модель програмування, яка може спростити код і покращити його читабельність. Тобто розробники можуть писати код, який описує поведінку програми, а не писати код, який визначає, як програма повинна поводитися. Це може полегшити розуміння коду, підтримку та налагодження. Ще реактивне програмування може покращити тестування програми. Оскільки основний наголос іде на відокремлення та компонування, розробники можуть писати більш модульний код, який легше тестувати. Це може зменшити складність процесу тестування та підвищити надійність програми.

Загалом, реактивне програмування є потужною парадигмою, яка може покращити продуктивність, швидкість реагування, стійкість, масштабованість і зручність обслуговування програм Java. Важливо також зазначити, що реактивне програмування не є заміною хорошій архітектурі та дизайну програмного забезпечення. Правильна архітектура та ПЗ все ще мають вирішальне значення для створення підтримуваних, масштабованих і розширюваних програм [8].

Висновки. Отже, використання реактивного програмування в Java-додатках є корисною та ефективною спробою підвищити продуктивність програм та покращити їхні функціональні можливості. Реактивне програмування надає розробникам потужний набір інструментів, які можна використовувати для створення високопродуктивних, адаптивних, стійких і

масштабованих програм. Одним із головних аспектів його застосування реактивного програмування в Java-додатках є перспектива швидко та легко адаптувати застосунки під значну кількість користувачів. Воно дає можливість програмісту просто додавати нові екземпляри моделі в застосунок та в рази збільшити число потоків для обробки запитів. Реактивне програмування в Java-додатках підвищує результативність користування ресурсами серверів. Адже при користуванні реактивними бібліотеками програмісти не лише оптимізують використання часу та пам'яті процесора, і цим зменшують навантаження на сервер, а й можуть реалізовувати мікросервіси з асинхронними та подійними інтерфейсами, що гарантує більшу гнучкість та ефективність мікросервісної архітектури. Це дає змогу знизити витрати на інфраструктуру та ефективніше використовувати ресурси.

Список інформаційних джерел

1. RxJava: Concurrency with RxJava, Reactor, and Akka Streams [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/329325452_RxJava_Concurrency_with_RxJava_Reactor_and_Akka_Streams.
2. Netty: сервер другого типу (Socket) [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://coderlessons.com/articles/java/netty-server-drugogo-tipa-socket.
3. Реактивне програмування [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F.
4. Знайомство з реактивними потоками – для Java-розробників [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/companies/piter/articles/353496/.
5. Difference between Hazelcast vs Ehcache [Електронний ресурс] – Режим доступу до ресурсу: https://www.educba.com/hazelcast-vs-ehcache/.
6. Cassandra vs MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/cassandra-vs-mongodb.
7. Паралелізм розподілених застосувань [Електронний ресурс] – Режим доступу до ресурсу: https://studopedia.com.ua/1_60873_paralelizm-rozpodilenih-zastosuvan.html.
8. Реактивне програмування на Java [Електронний ресурс] – Режим доступу до ресурсу: https://codeguida.com/post/1116.

УДК 004.021:004.9

Хоменко Олександр Миколайович, *аспірант кафедри інформаційних систем та технологій КПІ ім. Ігоря Сікорського, Україна, ORCID ID: https://orcid.org/0000-0003-1964-1097*
Науковий керівник: Гавриленко Олена Валеріївна, *кандидат фізико-математичних наук, доцент, доцент кафедри інформаційних систем та технологій КПІ ім. Ігоря Сікорського, Україна, ORCID ID: https://orcid.org/0000-0003-0413-6274*

КОНЦЕПЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ФОРМУВАННЯ ПОРТФЕЛІВ ПУБЛІЧНИХ (АДМІНІСТРАТИВНИХ) ПОСЛУГ

CONCEPT OF INFORMATION SYSTEM FOR FORMATION OF PUBLIC (ADMINISTRATIVE) SERVICES PORTFOLIOS

Анотація. У даній роботі описано схему роботи інформаційної системи для формування портфелів публічних (адміністративних) послуг. Розглянуто узагальнений процес роботи системи: збір, збереження, обробка даних, побудова аналітичних моделей для формування рекомендованих портфелів послуг на основі формату даних, взаємодія компонентів та опис учасників процесу. На основі структури вхідних даних зазначено методи, які можуть використовуватися для розв'язання поставленої задачі: статистичні методи, асоціативні правила, коефіцієнти близькості.
КЛЮЧОВІ СЛОВА: Інформаційна система, портфелі послуг, алгоритми, обробка даних, аналітичні моделі, статистичні методи, асоціативні правила, коефіцієнти близькості.

Abstract. This paper describes the scheme of the information system for the formation of portfolios of public (administrative) services. The generalized process of the system is considered: collection, storage, processing of data, construction of analytical models for the formation of recommended services portfolios based on the data format, interaction of components and description of process participants. Based on the structure of the input data, the methods that can be used to solve the given problem are specified: statistical methods, associative rules, similarity coefficients.
KEY WORDS: Information system, services portfolios, algorithms, data processing, analytical models, statistical methods, associative rules, similarity coefficients.

Вступ. Перехід до електронних послуг активно впроваджується в різних сферах діяльності людини. Вектор і прискорення розвитку цифрової трансформації визначається потребами людей, які стрімко змінюються в сучасному світі. Одним із напрямків цифрової трансформації є процес надання публічних адміністративних послуг, деякі із них доступні в електронному форматі. Але цей процес є нетривіальним, комплексним і породжує набір задач, які виникають в процесі покращення існуючих сервісів. Об'єднання послуг за певним критерієм у портфель послуг – одна із задач. Для більш якісного впровадження електронних послуг та проведення реінжинірингу послуг є актуальним формування наборів послуг, які будуть називатися портфелем або групою послуг [1,2]. Для вирішення проблеми формування портфелів послуг (груп послуг) потрібно створити інформаційну систему, яка могла б на основі вхідних даних формувати рекомендації Клієнту для покращення процесу надання електронних послуг. Користувач замовляє деякі послуги і за допомогою інформації про послуги, аналізу даних встановлюються зв'язки між послугами і формується набір рекомендованих послуг – портфель, який буде йому корисним. В даній роботі здійснюється огляд моделі інформаційної системи для формування портфелів публічних адміністративних послуг.

Що таке інформаційна система? Існує декілька визначень терміну «Інформаційна система», які формуються в залежності від різних чинників. В області інформаційних технологій термін можна сформулювати наступним чином: Інформаційна система – система, яка використовує комп'ютерне обладнання та програмне забезпечення, бази даних, моделі аналізу, планування, контролю та прийняття рішень. У визначенні не заперечується важливість інших характеристик інформаційної системи, але виділяється важливість технології у формі апаратного забезпечення, мереж і програмного забезпечення над іншими аспектами [3]. Більш узагальнене визначення: система, призначена для збору, обробки, зберігання та розповсюдження інформації. Виділяють шість компонентів, які разом визначають інформаційну систему: апаратне забезпечення, програмне забезпечення, дані, процедури, люди, Інтернет (не є необхідним) [4].

Архітектура та схема роботи інформаційної системи. Систему можна розділити на декілька рівнів: інтерфейс для інтеграції із зовнішніми системами, обробка даних, збереження даних, аналіз та візуалізація даних, логування та моніторинг роботи системи.

Схема роботи інформаційної системи для вирішення поставленої задачі, зображена на рисунку 1.

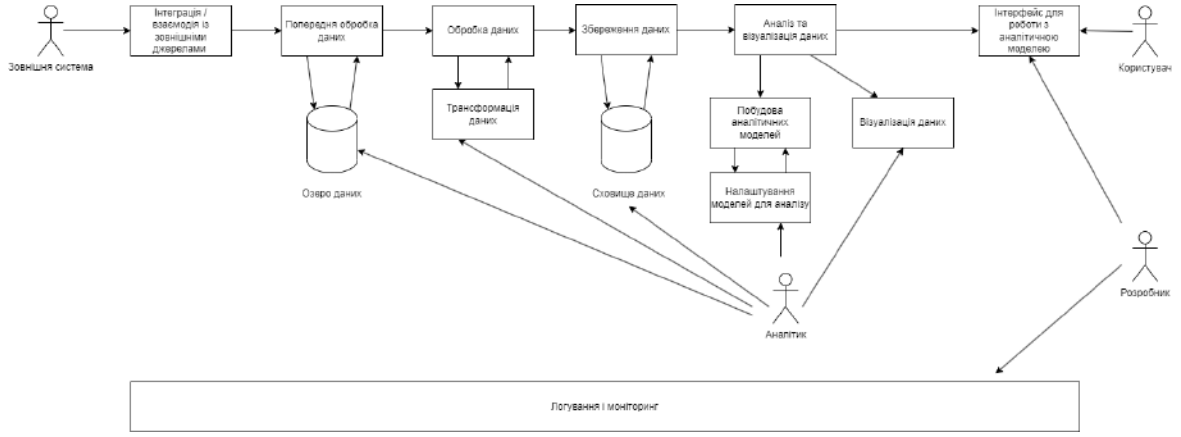


Рис. 1. Архітектура інформаційної системи для формування портфелів публічних адміністративних послуг

Збір даних для аналізу. Користувач або зовнішня система надсилають дані до інформаційної системи через визначений інтерфейс взаємодії. Існують проблеми з джерелом даних, у Користувача можуть бути різні формати даних відповідно до предметних областей. Необхідно провести анкетування, щоб по критеріях можливо було б визначити рекомендаційні алгоритми і процес їх роботи.

Типи вхідних даних:

1) Транзакція 1: Послуга1, ... , Послуга n. Цей формат даних може бути використаний для побудови асоціативних правил. Приклад даних наведений у таблиці 1.

2) Послуга 1: Документ1,…, Документ n. Цей тип даних може використовуватися для обчислення коефіцієнтів близькості. Приклад даних наведений у таблиці 2.

3) Послуга 1 – День (дата) – Кількість замовлень. За допомогою статистичних методів можна встановити зв’язки між послугами та сформувати портфелі послуг. Приклад даних наведений у таблиці 3.

Кількість форматів вхідних даних може збільшуватися.

Табл. 1. Приклад даних для побудови асоціативних правил

№ транзакції	Перелік послуг, які замовляв користувач у транзакції
1	Послуга 1, Послуга 2, Послуга 3

Табл. 2. Приклад даних для обчислення коефіцієнтів близькості

№	Назва послуги	Перелік документів, які необхідні для отримання даної послуги
1	Послуга 1	Документ 1, документ 2, документ 3

Табл. 3. Приклад статистичних даних для формування портфелів послуг

	День 1	День 2	День 3	День 4	День 5
Послуга 1	1	0	1	0	1

Попередня обробка даних. Система обробляє отримані дані і зберігає їх у сховищі даних. Дані у вхідному форматі зберігаються у Data Lake. Аналітик даних може проводити попередній аналіз, візуалізацію даних для покращення аналізу, аналітичних моделей - визначає структуру даних, можливість застосування алгоритмів для вирішення поставлених задач.

Одним із етапів є обробка відсутніх значень у вхідних даних. Деякі аналітичні моделі працюють некоректно з відсутніми значеннями у вхідних даних. Існує кілька методів обробки таких ситуацій і вибір методу залежить від предметної області, мети і способу використання даних в аналізі. Найпростіше рішення у цій ситуації є застосування константного значення для заповнення відсутніх значень або ігнорування цих рядків (потрібно дивитися чи не зменшиться якість даних для аналізу), але цей метод використовується, коли відсутні значення важко спрогнозувати. Альтернативний варіант – застосувати середнє значення або медіану для заповнення відсутнього значення по вказаному атрибуту. Інший варіант - алгоритм інтелектуального аналізу даних, щоб передбачити ймовірне значення.

Обробка даних. На цьому етапі відбувається фільтрування та перетворення даних. У вхідній вибірці можуть міститися шуми, які були створені через помилки введення даних або через помилки передачі даних тощо. Шуми можна обробити за допомогою binning. Згладжування може здійснюватися за середнім значенням, медіаною або межами діапазону. Схожа процедура застосовується для оброблення викидів. Інший механізм для визначення груп викидів даних - кластеризація. Виявлені викиди можна згладити або видалити. Аномалії в даних можна виявити за допомогою візуального аналізу, графіків.

Інше питання, яке може виникнути при обробці даних - рядки, які дублюються або мають нерелевантні значення. Ці дані потрібно видалити, бо можуть негативно вплинути на якість аналітичної моделі - зменшиться якість результатів, а час роботи алгоритму може збільшитися. Оброблені дані зберігаються у визначеному форматі в сховищі даних для подальшого аналізу.

Побудова аналітичних моделей. Моделі для аналізу оновлюються при надходженні в обробку нових даних. Графіки, діаграми полегшують процес аналізу результатів. На поточний момент для побудови аналітичної моделі використовуються такі алгоритми:

- *Статистичні методи* [1]: побудова матриці парних коефіцієнтів кореляції; обчислення таблиці коефіцієнтів критерія Стьюдента; формування рекомендацій для створення портфелів послуг.
- *Асоціативні правила* [2]: застосування алгоритму Apriori для виявлення побудови асоціативних правил; визначення ланцюгів послуг, використовуючи транзитивність

відношень; перетворення ланцюгів послуг в рекомендовані портфелі послуг, враховуючи мінімальний confidence.

- *Коефіцієнти близькості*: формування матриць попарних коефіцієнтів подібності (Жаккара, Кульчицького, Браун-Бланке та Отіаї); формування множини подібних послуг; формування рекомендацій для створення портфелів послуг.

Логування і моніторинг роботи системи. Цей етап відіграє важливу роль в будь-якій системі, бо збір і обробка інформації є корисною для розробників. Метрики використовуються для моніторингу роботи системи і її роботи. Доступ до компонентів системи здійснюється відповідно до ролей користувачів. Аналітики мають можливість взаємодіяти з моделями, сховищами даних для виконання певного спектру задач. Розробники перевіряють роботу системи за допомогою метрик, покращують процеси, аналізують і реагують на певні ситуації.

Взаємодія з системою. Аналітична модель доступна через визначений інтерфейс API для взаємодії з Користувачами / інтеграції з системами, які мають встановлений доступ.

Учасники в системі. В інформаційній системі було виділено наступні учасники, які зображені на рисунку 1: Зовнішня система, Користувач, Розробник, Аналітик. Зовнішня система – джерело даних для проведення аналізу і побудови аналітичних моделей. Надсилає дані до інформаційної системи, яка обробляє їх і проводить аналіз. Користувач – людина, яка робить запит про отримання рекомендованих портфелів послуг на основі її запиту – множини послуг. Розробник – людина, що розробляє, оновлює систему, має доступ до всіх компонентів. Аналітик – людина, що аналізує дані, процес обробки даних і роботу аналітичних моделей.

Висновки. Описана архітектура системи задовольняє критерії для інформаційної системи. Для роботи системи потрібно апаратне забезпечення (сервер), програмне забезпечення (інструменти / бібліотеки для обробки даних), дані (статистична інформація про послуги), процедури (набір інструкцій для роботи компонентів системи і їх взаємодії), люди (Користувачі, зовнішні системи), Інтернет (взаємодія кінцевих користувачів з системою).

У цій роботі було описано архітектуру для побудови інформаційної системи для формування портфелів публічних адміністративних послуг. Було наведено приклад роботи системи і опис, чому ця модель є інформаційною системою.

Список інформаційних джерел

1. Gavrilenko, O., Zhurakovska, O., Kohan, A., Matviichuk, R., Piskun, A., Khavikova, Y. and Khalus, O. (2022), “The principle for forming a portfolio of public services based on the analysis of statistical information”,/ Eastern-European Journal of Enterprise Technologies, No 3 (3 (117)), pp. 57–64, doi: https://doi.org/10.15587/1729-4061.2022.260136.

2. Gavrilenko , O. ., Khomenko , O. ., Zhurakovska , O. ., Kohan , A. ., Piskun , A. ., & Khalus , O. . (2022). APPLICATION OF ASSOCIATION RULES FOR FORMATION OF PUBLIC (ADMINISTRATIVE) SERVICES PORTFOLIO. Advanced Information Systems, 6(4), 63–68. https://doi.org/10.20998/2522-9052.2022.4.09

3. Sebastian K Boell, Dubravka Cecez-Kecmanovic (2015), “What is an Information System?”, Conference: Proceedings of the 48th Hawaiian International Conference on System Sciences (HICSS). DOI:10.1109/HICSS.2015.587.

4. Information system [Електронний ресурс] - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Information_system.

УДК 004.416

Глушко Богдан Сергійович, *здобувач вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник: Баклан Ігор Всеволодович, *кандидат технічних наук,*
доцент, доцент кафедри інформатики та програмної інженерії
КПІ ім. Ігоря Сікорського, Україна

ДОМЕННО-ОРІЄНТОВАНА МОВА ДЛЯ ФРАКТАЛЬНОГО АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

Анотація. Застосування часових рядів для аналізу та прогнозування тих чи інших явищ продовжує бути одним із популярних засобів сучасної аналітики, бізнес-планування, трейдингу, окремих напрямів економіки та багатьох інших аспектів сучасних прикладних наук та професій. Протягом останніх двадцяти років, з-поміж засобів аналізу та дослідження, особливе місце стали займати фрактальні методи аналізу, з огляду на їх значну точність та зручність використання. Втім, актуальною залишається проблема важкої доступності інструментів для аналізу та їх високий поріг входу. Результатом проведеної роботи стало створення макрос-базованої мови програмування для аналізу та обробки часових рядів, а також інтегроване середовище розробки із можливістю масштабування для подальшого розвитку системи.
КЛЮЧОВІ СЛОВА: фрактальний аналіз часових рядів, системний аналіз та прогнозування, методи фрактального прогнозування.

Abstract. The use of time series for analysis and forecasting of various phenomena continues to be one of the popular means of modern analytics, business planning, trading, specific areas of economics, and many other aspects of modern applied sciences and professions. Over the past twenty years, fractal analysis methods have become increasingly important in analysis and research, due to their high accuracy and ease of use. However, the problem of difficult accessibility of tools for analysis and their high entry threshold remains relevant. The result of the work carried out was the creation of a macro-based programming language for time series analysis and processing, as well as an integrated development environment with scaling capabilities for further system development.
KEY WORDS: fractal analysis of time series, system analysis and forecasting, fractal forecasting methods.

Вступ. У сучасному світі збільшення об'єму та складності даних є нормою для бізнесу, фінансів та наукових досліджень. Одним із ефективних способів аналізу та використання таких даних є аналіз часових рядів, який полягає у вивченні та прогнозуванні залежності між подіями, що відбуваються в часі. Однак, збільшення обсягу та складності даних також створює виклики для аналітиків даних та дослідників, оскільки вони повинні знайти нові та більш ефективні методи ефективного виконання своєї роботи.

Зважаючи на значущість аналізу та прогнозування часових рядів, проблематикою виконання роботи було обрано тему розробки доменно-орієнтованої мови програмування для фрактального аналізу та прогнозування часових рядів. Дана тема має високу актуальність у сучасному світі, де зростає обсяг та складність даних, і необхідно розробляти більш ефективні та потужні інструменти для їх подальшого використання.

В рамках роботи виконано аналіз існуючих методів аналізу та прогнозування часових рядів, включаючи методи звичайної статистики, аналізу спектра, аналізу хвильових процесів, фрактального аналізу та методів машинного навчання. В контексті

поставленої задачі, фрактальний аналіз є одним з найбільш ефективних методів аналізу та прогнозування часових рядів, оскільки він дозволяє вивчати нелінійні та фрактальні характеристики часових рядів. Цей метод може бути використаний для прогнозування майбутніх значень часових рядів та виявлення складних структур та залежностей в даних.

Однак, існуючі інструменти для фрактального аналізу та прогнозування часових рядів мають деякі недоліки. Наприклад, вони можуть бути складними у використанні або не дозволяти працювати зі специфічними даними, що характерні для деяких доменів. Також, існуючі інструменти можуть бути недостатньо ефективними для вирішення складних задач, які вимагають використання різних методів аналізу та прогнозування.

Як результат, було розроблено доменно-орієнтовану мову програмування для фрактального аналізу та прогнозування часових рядів із власним інтегрованим середовищем розробки. Створений продукт дозволив спростити процес аналізу часових рядів, і застосувати значну кількість оновлених компонентів для подальшої обробки отриманих значень.

Основна частина. Безпосереднє виконання роботи було поділено на декілька ключових компонентів, що дозволили декомпозувати поставлену задачу на спрощені частини роботи. Першою частиною став аналіз існуючих методів, із включенням порівняльного аналізу із фрактальними методами, що надають ідентичні результати. Було додано засоби перевірки швидкості відпрацювання методів та порівняльний аналіз. Другою складовою роботи стало створення безпосередньо мови програмування, що включила в себе необхідні компоненти аналізу вхідного потоку даних, проміжний обробник інформації, системи аналітики синтаксису та інтепретативного запуску складових. Нарешті, останнім компонентом стала система побудови графіків та перевірки моделі на її валідність. Поєднання цих аспектів дозволило отримати фінальний продукт, що готовий до використання та подальшого впровадження у комплексні аналітичні проекти.

Розробка макрос-базованої мови програмування. Створення макрос-базованої мови програмування для аналізу та обробки часових рядів почалося з визначення вимог до системи та формування її архітектури. Оскільки метою системи було забезпечити зручний та ефективний аналіз та прогнозування часових рядів, було визначено, що на першому етапі потрібно розробити синтаксичний аналізатор та інтерпретатор макросів. Для розробки системи інтерпретації була використана мова програмування Python, з огляду на можливість перевикористати інтерпретативні модулі та забезпечити проміжну обробку вхідних даних у спрощеній формі. Створена система для аналізу та прогнозування була розбита на декілька модулів, кожен з яких виконує конкретну функцію.

Першим модулем є модуль для збору даних. Цей модуль має на меті збирати та зберігати дані відповідно до заданих критеріїв, таких як часовий інтервал, тип даних, масштаб та інші параметри. Модуль забезпечує автоматичне збирання даних з веб-сторінок, баз даних та інших джерел даних.

Другим модулем є модуль для попередньої обробки даних. Цей модуль має на меті очищення та перетворення даних для подальшого аналізу. Він може включати в себе функції для видалення відсутніх даних, відфільтрування шуму та інших аномалій, а також для масштабування даних відповідно до потреб користувача.

Третім модулем є модуль для аналізу даних. Цей модуль має на меті проведення різних видів аналізу даних, таких як частотний аналіз, аналіз складових, фрактальний аналіз та інші. Він включає в себе різні методи та алгоритми для обробки даних, що дозволяють проводити детальний аналіз.

Четвертим модулем є модуль для прогнозування. Цей модуль має на меті створення прогнозу на основі аналізу даних

та їхньої історії. Він включає в себе різні методи та алгоритми прогнозування, такі як методи залежності від масштабу, методи фрактального аналізу та інші.

Основним аспектом аналітичної компоненти мови було додавання модуля для синтаксичного аналізу макросів, який було реалізовано з використанням регулярних виразів. Він забезпечує перевірку правильності синтаксису макросів та розбиття їх на синтаксичні одиниці, що далі передаються в інтерпретатор.

Інтерпретатор було реалізовано використовуючи патерн "Команда", що дозволяє зберегти макроси у вигляді об'єктів команд, які можуть бути виконані під час інтерпретації. Кожен макрос має свій власний об'єкт команди, який містить інформацію про функцію, яку треба виконати, та аргументи, які потрібні для її виконання. Цей підхід дозволяє легко додавати нові макроси та розширювати функціонал системи.

Система графічного відображення ряду. Процес розробки та інтеграції модуля візуалізації часових рядів був важливою складовою розробленої системи. Модуль візуалізації було розроблено з метою забезпечення користувачам можливості відображення та аналізу результатів аналізу часових рядів у зручному та інтуїтивно зрозумілому вигляді.

У процесі розробки модуля було використано мову програмування Python та бібліотеки matplotlib та pandas для побудови графіків. Було розроблено власний алгоритм масштабування побудованих графіків часових рядів, який дозволяє побудувати графік з точністю до масштабу умовної одиниці. Це забезпечує користувачам можливість підглядати деталі на графіках часових рядів та здійснювати аналіз на більш високому рівні деталізації.

Для забезпечення більш ефективного та зручного відображення графіків часових рядів була реалізована адаптивна система обробки даних. Це дозволяє системі адаптуватися до обсягу даних, а також дозволяє користувачам змінювати параметри відображення графіків залежно від їх потреб. Окремо слід зазначити про систему прогнозованої візуалізації. Ця система дозволяє відображати прогнозовану поведінку часового ряду враховуючи збереження стаціонарності. Це забезпечує користувачам можливість побачити, як може змінитися поведінка часового ряду в майбутньому та зробити прогноз на основі цих даних.

Для досягнення мети було розроблено власний алгоритм масштабування графіків, що дозволяє користувачам більш детально проаналізувати динаміку часових рядів на різних масштабах, а також динамічно змінювати масштабування в залежності від потреб користувача. Цей алгоритм забезпечує коректну візуалізацію на різних масштабах, що є важливим фактором при аналізі часових рядів.
Також було реалізовано адаптивну систему обробки даних, що дозволяє відображати графіки в режимі реального часу з урахуванням зміни параметрів часового ряду, таких як середнє значення, дисперсія, кореляція та інші параметри. Ця система дозволяє користувачам детально аналізувати динаміку часових рядів та прогнозувати їх поведінку в майбутньому.

Окремо було реалізовано систему прогнозованої візуалізації, яка враховує збереження стаціонарності часового ряду при прогнозуванні його поведінки в майбутньому. Ця система дозволяє користувачам детально аналізувати динаміку часового ряду та його прогнозовану поведінку на основі передбачення можливості збереження властивості стаціонарності.

Інтегроване середовище розробки. Складовою, що об’єднала усі проектні компоненти у єдиний фіналізований продукт, стала розробка інтегрованого середовища розробки. Цей компонент дозволив об’єднати усі системи обробки, проміжні модулі та алгоритмічний інтерпретатор у єдиний компонент, зручний для роботи фінальному

користувачу. Загалом, схему об'єднання системи можна подати у наступному вигляді:

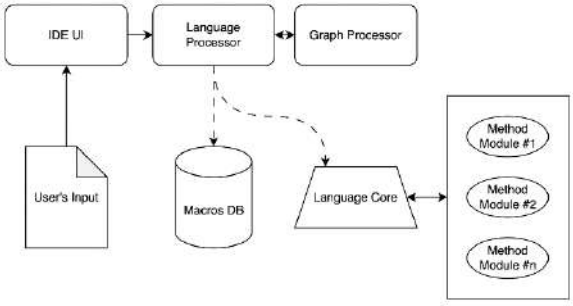


Рис.1. Система композиції функціональних модулів

Як показано на малюнку, саме інтегроване середовище розробки приймає вхідні дані від користувача, надсилаючи їх далі у мовний обробник. Наступним кроком є порівняння отриманих даних із базою макрос-команд, а також надсилання двосторонніх запитів до модуля візуалізації. Нарешті, після отримання співпадіння команди макроса, дані надсилаються на мовне ядро, що викликає необхідний модуль обробки. Отримані дані передаються назад у графічний процесор, а через нього - в Інтегроване Середовище Розробки.

Висновки. У даній роботі були розглянуті різні методи аналізу та прогнозування часових рядів, зокрема методи спектрального аналізу, хвильового аналізу, фрактального аналізу, методи гуртків, методи залежності від масштабу та метод масштабування. Було показано, що використання часових рядів є важливим у сучасному бізнесі та економіці для прогнозування та планування, а також для досліджень у прикладних науках. Особлива увага приділена фрактальним методам аналізу та прогнозування, оскільки вони забезпечують високу точність та зручність використання. Проте, однією з проблем використання фрактальних методів є важка доступність інструментів для аналізу та високий поріг входу. Для розв'язання цієї проблеми, було розроблено макрос-базовану мову програмування для аналізу та обробки часових рядів, а також інтегроване середовище розробки з можливістю масштабування. Додатково, був розроблений модуль візуалізації часових рядів, який включає в себе алгоритм масштабування та систему прогнозованої візуалізації. Отже, результатом проведеної роботи стала розробка комплексу засобів для аналізу та прогнозування часових рядів, що може знайти своє застосування в різних галузях науки та бізнесу.

Список інформаційних джерел

1. Shumway, R. H., & Stoffer, D. S. (2010). Time series analysis and its applications: With R examples. Springer Science & Business Media.
2. Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (2008). Time series analysis: forecasting and control. John Wiley & Sons.
3. Brockwell, P. J., & Davis, R. A. (2016). Introduction to time series and forecasting (3rd ed.). Springer.
4. Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice (2nd ed.). OTexts.
5. Granger, C. W. J., & Newbold, P. (1974). Spurious regressions in econometrics. Journal of Econometrics, 2(2), 111-120.
6. Mandelbrot, B. (1975). Les objets fractals: forme, hasard et dimension. Flammarion.
7. Gabaix, X. (2009). Power laws in economics and finance. Annual Review of Economics, 1(1), 255-294.
8. Turchenko, M., & Stepchuk, O. (2019). A comparative analysis of fractal-based methods for stock market time series forecasting. Future Internet, 11(4), 90.
9. Fan, J., & Gijbels, I. (1996). Local polynomial modelling and its applications: monographs on statistics and applied probability. Chapman and Hall.
10. Mallat, S. (2009). A wavelet tour of signal processing: The sparse way. Academic Press.

УДК 004.891

***Майборода Аріна Миколаївна**, здобувачка вищої освіти,*
КПІ ім. Ігоря Сікорського, Україна
***Науковий керівник: Солдатова Марія Олександрівна**,*
доцент кафедри інформаційних систем та технологій,
КПІ ім. Ігоря Сікорського, Україна

СЕГМЕНТУВАННЯ СПОЖИВАЧІВ ДЛЯ ЯКІСНОГО МАРКЕТИНГУ У СОЦІАЛЬНИХ МЕРЕЖАХ

Анотація. Задача та проблематика просування контенту в соціальних мережах за останні кілька років набула неабиякої популярності та важливості. Саме через це маркетинг у соціальних мережах вимагає якісного сегментації споживачів.
КЛЮЧОВІ СЛОВА: СЕГМЕНТАЦІЯ, КЛАСТЕРИЗАЦІЯ, МЕТОД ГРАДІЄТНОГО СПУСКУ, МЕТОД ЛІКТЯ

Abstract. The task and issues of content promotion in social networks have gained considerable popularity and importance over the past few years. It is because of this that marketing in social networks requires high-quality segmentation of consumers.
KEY WORDS: SEGMENTATION, CLUSTERIZATION, GRADIENT Descent METHOD, ELBOW METHOD

Вступ. Задача та проблематика просування контенту в соціальних мережах за останні кілька років набула неабиякої популярності та важливості. З подальшим розвитком автоматизації процесів, виникає питання, чи можливо здійснювати якісну рекламну кампанію за допомогою стратегії, яка була створена, використовуючи результати прогностичних моделей та моделей кластеризації? Щоб мінімізувати втручання суб'єкта, тобто фахівця з маркетингу у соціальних мережах, в процес створення плану просування. Ми відповідаємо на це питання так.

Це дозволяє налаштувати ефективну рекламну кампанію та забезпечити максимальний результат від просування продукту або послуги. Оптимальна сегментація дозволяє підібрати цільову аудиторію, зважаючи на її поведінку та інтереси в соціальних мережах. Це забезпечує зростання продажів та популярності бренду у соціальних медіа. Аналогів, в яких є автоматизоване просування контенту досить небагато, це пояснюється відсутністю достатньої бази знань про формування стратегії просування, які використовують прогностичні моделі та моделі кластеризації. Одним з найпопулярніших є Audiense — це програмне забезпечення для соціального медіа маркетингу, розроблене, щоб допомогти користувачам максимізувати покази своїх облікових записів в соціальних мережах, таких як Twitter, з цілею популяризації. Це дозволяє їм ефективніше взаємодіяти зі своєю аудиторією, надаючи їм інструменти для аналізу даних, зібраних із цих сайтів соціальних мереж. Також, існує досить популярна платформа StackAdapt — це рекламна платформа, яка допомагає агентствам і брендам охоплювати та шукати своїх ідеальних клієнтів за допомогою привабливих блоків нативної та відеореклами. [1]

Предмет та мета дослідження

Предметом дослідження є маркетинг у соціальних мережах та його складові. Метою

дослідження є автоматизація просування контенту у соціальних мережах для підвищення ефективності працездатності фахівця з маркетингу. В рамках даної статті ми розглянемо першу важливу частину для якісного просування контенту – створення цільових аудиторій. Для цього необхідно виконати такі задачі:

- виконати огляд існуючих методів створення цільових аудиторій;
- сформулювати математичну постановку задачі;
- здійснити порівняльний аналіз методів, які використовуються при сегментації споживачів.

Задача створення цільових аудиторій. Цільова аудиторія – це група людей, визначених як ймовірні клієнти компанії. Вони об'єднані між собою спорідненими рисами. На цьому етапі ми будемо розподіляти користувачів на сегменти.
Вхідні дані: люди, їх вік, їх інтереси.

Вихідні дані: цільові аудиторії — сегменти розподілені за віком та інтересами людей.

Алгоритм розв'язання задачі створення цільових аудиторій зображений на рисунку 1.

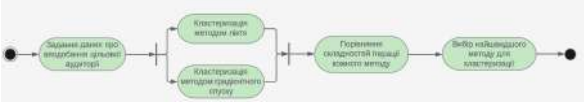


Рисунок 1 – Алгоритм розв'язання задачі створення цільових аудиторій

Розглянемо більш детально постановку даного алгоритму. Цільові аудиторії формуємо за допомогою кластеризації. На вхід подається множина об'єктів. На виході отримуємо об'єкти, розміщені по кластерам. Для розв'язання даної задачі будемо використовувати метод кластеризації k-means.

Мета методу — розділити n спостережень на k кластерів, так щоб кожне спостереження належало докластера з найближчим до нього середнім значенням. Метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера, тобто функції 1.

$$\sum_{i=1}^{N} d\left(x_i, m_j\left(x_i\right)\right)^2, (1)$$

де:

- d – метрика;
- x_i – і-тий об'єкт даних;
- $m_j\left(x_i\right)$ – центр кластера, якому на j -тій ітерації приписани1 елемент x_i.

Головні переваги методу k-середніх — його простота та швидкість виконання.
Особливістю даного методу є те , що на вхід має подаватись кількість кластерів, які ми визначаємо наступними методами її розв'язання: методом градієнтного спуску та методом ліктя. Потім при порівнянні даних методів, ми оберемо найкращий та за ним буде створена цільова аудиторія для майбутнього просування контенту.
Основною задачею методу кластеризації у нашому випадку є сегментація цільової аудиторії на підмножини. У нашому випадку, це категорії (кулінарія, медицина, IT, і т.д.). Тому нашою задачею є сегментація на кластери цільової аудиторії за категоріями. Оскільки в базі даних є інформація про їхні вподобання категорій, а їх може бути кілька, то сегментація буде відбуватись по найулюбленіший категорії цієї людини.

Кластеризація методом k-means з визначенням кількості кластерів методом градієнтного спуску

Градієнтний спуск – ітераційний алгоритм оптимізації, який дозволяє мінімізувати функціонал помилки шляхом оновлення параметрів шуканої функції у зворотному напрямку градієнта. [6]
Ідея градієнтного спуску: на кожному кроці оновлювати параметри шуканої функції у зворотному напрямку градієнта функціоналу помилки, тим самим здійснюючи оптимізацію у напрямку найшвидшого спуску.

Для пошуку мінімуму множини кластерів функції k-means J(C) використовується метод градієнтного спуску.

Математична постановка задачі

Для вибору числа кластерів використовується значення функції k-means J(C) . Обирають те число кластерів, починаючи з якого значення J(C) спадає не так швидко.

Введемо наступні позначення:

- $C = \left\{\vec{C_1}, \vec{C_2}, ..., \vec{C_k}\right\}$ — множина центроїдів;
- $X = \left\{\vec{X_1}, \vec{X_2}, ..., \vec{X_n}\right\}$ — множина даних, яку необхідно кластеризувати;

Розглянемо функцію класичного алгоритму k-means. Вона наведена у формулі 2.

$$J(C) = \sum_{k=1}^{C \vee \sum_{i=1}^{|X_k|} \sum_{j=1}^{n} \left(x_{i,j} - C_{k,j}\right)^2} \quad (2)$$

Далі потрібно мінімізувати суму квадратів Евклідової відстані елементів кластера від відповідного центроїда. Представлення мінімізації наведене у формулі 3.

$$J(C) = \sum_{k=1}^{|C|} \sum_{i=1}^{|X_k|} f^2\left(\vec{X_i}, \vec{C_k}\right) (3)$$

Знайдемо похідну нульової функції. Функція наведена у формулі 4. [2]

$$\frac{\partial J(C)}{\partial C_{a,b}} = \sum_{i=1}^{|X|} \frac{\partial min |C| \, j=1 \, f^2\{\vec{x_i}, \vec{c_j}\}}{\partial C_{a,b}} (4)$$

Кластеризація методом k-means з визначенням кількості кластерів методом ліктя

При кластеризації методом k-середніх кількість кластерів найчастіше оцінюють за допомогою «методу ліктя». Він передбачає багаторазове циклічне виконання алгоритму зі збільшенням кількості кластерів, що вибираються, а також подальшим відкладанням на графіку бала кластеризації, обчисленого як функція від кількості кластерів. Основним показником методу ліктя є $J(C)$ - сума квадратів помилок. [7]

Основна ідея методу ліктя полягає в тому, що в міру збільшення числа кластерів k розподіл вибірок буде більш точним, а ступінь агрегації кожного кластера поступово збільшуватиметься, тому квадрат помилок, природно, поступово стануть меншими. Більше того, коли k менше числа істинних кластерів, збільшення k значно збільшить ступінь агрегації кожного кластера, тому зниження $J(C)$ буде більшим, і коли k досягне числа істинних кластерів, значення, отримане шляхом збільшення k. Ступінь агрегації, повертається швидко стає менше, тому зниження $J(C)$ буде різко зменшуватися, а потім поступово згладжуватиметься, оскільки значення k продовжує збільшуватися, що означає, що відносини між $J(C)$ і k мають форму коліна, і це коліно Відповідне значення k є кількістю справжніх кластерів у даних. Звичайно, саме тому метод називається методом ліктя. [7]

Математична постановка задачі

Математична постановка задачі полягає у мінімізації множини кластерів функції витрат k-means та наведена у формулі 6.

$$J(C) = \sum_{k=1}^{K} \sum_{i \epsilon C_k} \left|\left|x_i - \mu_k\right|\right|^2 \rightarrow min, \; C \; (6)$$

Де:

C – множина кластерів потужності K

μ_k – центроїд кластера C_k

Для вирішення питання вибору кількості кластерів потрібно знайти мінімум, але оскільки такий мінімум буде при тому, що кожна точка – це кластер одного елементу, то потрібно скористатись правилом, побудувавши графік, знайти точку, де J(C) падає не так швидко. Дане представлення наведене формулою 7.

$$D(k) = J\left(C_k\right) - J\left(C_{k+1}\right) \vee \frac{}{J\left(C_{k-1}\right) - J\left(C_k\right) \vee = \rightarrow min, \; C \; (7)}$$

Візьмемо частину нашої цільової аудиторії та побудуємо графік, аби визначити оптимальну кількість кластерів для такої кількості людей. Це показано на рисунку 2.

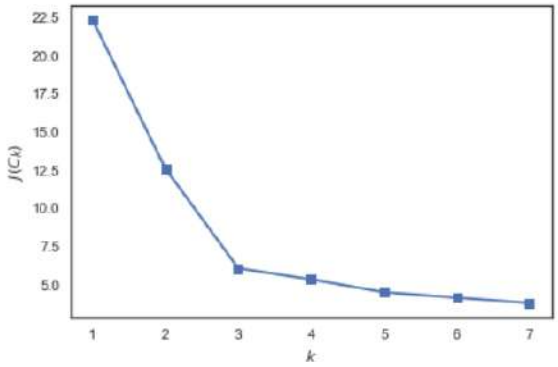


Рисунок 2 – Графік побудови методу ліктя

На рисунку бачимо, що при зміненні k від 3 до 4, значення J(C) падає не так швидкому, тому для цього варіанту оптимальною кількістю кластерів буде 3 кластери.

Порівняння методів ліктя та градієнтного спуску

Для порівняння методів пошуку кількості кластерів проведемо експеримент. Отже, спочатку візьмемо таблицю даних можливих інтересів, для використання у подальшому обрахунку. Це показано на рисунку 3.

Results | Messages

	Id	Name
1	1	Кулінарія
2	2	Медицина
3	3	IT
4	4	Ігрова сфера
5	5	Кіно
6	6	Театр
7	7	Фітнес та спортзал
8	8	Крипто-сфера
9	9	Бізнес
10	10	Туризм
11	11	Мистецтво

Рисунок 3 – Таблиця інтересів

Оскільки всього у нас вікових категорій три, то за максимального порогу ми отримаємо 33 кластери. Візьмемо інформацію з бази даних та виконаємо кластеризацію з пошуком кількості кластерів методом ліктя та методом градієнтного спуску.

Результати відображені у вигляді кривих на графіках на рисунках 4 та 5.

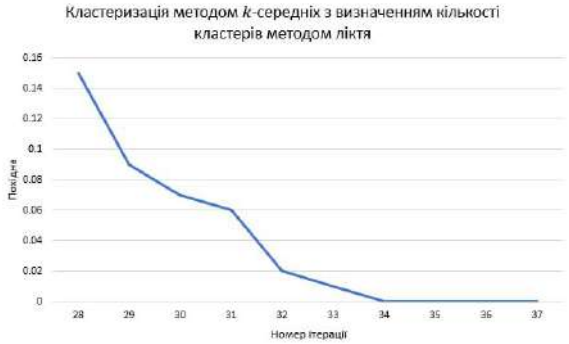


Рисунок 4 – Кластеризація методом k-середніх з визначенням кількості кластерів методом ліктя

Для методу ліктя бачимо, що починаючи з 34-ої ітерації, похідна дорівнює нулю. Але так як наша максимальна кількість кластерів 33, то обираємо 33. Якщо кожна людина належить до певної вікової категорії та має свої інтереси, то вона може бути віднесена до конкретного кластеру з цими ж характеристиками, тому людей, які не потрапили в кластер, немає. Отже, оскільки метрика Precision становить 100%, ми можемо стверджувати, що точність дорівнює 100%. Це означає, що значення метрики Recall буде дорівнювати 0.

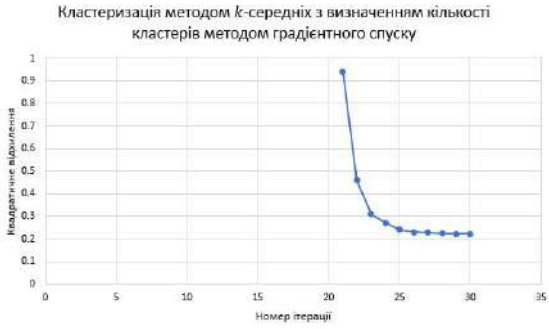


Рисунок 5 – Кластеризація методом k-середніх з визначенням кількості кластерів методом градієнтного спуска

Для методу градієнтного спуску бачимо, що починаючи з 24-ої ітерації, квадратичне відхилення падає не так швидко, як раніше. Розглянемо два варіанти, де візьмемо кількість кластерів 25 та 30.

Для 25 кластерів дей, які не потрапили в кластер, дорівнює 25% та для 30 кластерів буде дорівнювати 10%. Отже, чим більше кластерів – тим більше точність

кластеризації. Вони відрізняються Precision, у першому варіанті Precision складає 75%, а в другому 90%. Таким чином Recall дорівнює 25% та 15% відсотків відповідно.

Отже, висновок полягає в тому, що використання методу градієнтного спуску призведе до більшої повноти кластерів (більше людей буде в кожному кластері), але може привести до втрати приорітетності інтересів. З іншого боку, метод ліктя забезпечує найкращу точність розрахунків, що особливо важливо для кластеризації цільової аудиторії. Таким чином, обираючи метод ліктя, користувач матиме можливість обрати цільову аудиторію з найбільшим інтересом саме до цієї теми.

Висновок. В даній роботі проведено дослідження в рамках сегментації споживачів для створення якісної рекламної кампанії. Розподілення людей визначено за допомогою кластеризації методом k-середніх. Також, ми дізналися, що даний метод потребує на вхід очікувану кількість кластерів. Її ми знаходимо двома методами: методом ліктя та методом градієнтного спуску. Після чого ми обрали найкращий та прийшли до висновку, що метод ліктя дозволяє це зробити швидше та з меншою похибкою для великого об'єму даних. Описаний підхід допоможе експертам у галузі маркетингу значно полегшити процес складання плану просування контенту, створення якісної рекламної кампанії.

Список інформаційних джерел

1. «Audiense» review. – 2022. – [Електронний ресурс]. – Режим доступу до ресурсу: https://crozdesk.com/software/audiense.
2. Solution of the clustering problem by the method of gradient descent. 2022. – [Електронний ресурс]. – Режим доступу до ресурсу: https://habr.com/ru/articles/188638/.
3. Кластеризація методом k-середніх. – 2022. – [Електронний ресурс]. – Режим доступу до ресурсу:
4. https://www.wiki-data.uk-ua.nina.az/Кластеризація_методом_к–середніх.html
5. Метод ліктя. – 2022. – [Електронний ресурс]. – Режим доступу до ресурсу: https://www.helenkapatsa.ru/mietod-loktia/
6. Розв'язування задачі кластеризації методом градієнтного спуску. – [Електронний ресурс]. – Режим доступу до ресурсу: https://habr.com/ru/post/188638/
7. Градієнтний спуск. – [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D1%96%D1%94%D0%BD%D1%82%D0%BD%D0%B8%D0%B9_%D1%81%D0%BF%D1%83%D1%81%D0%BA
8. Кваліфікаційна робота на тему "Дослідження методів кластеризації часових рядів за значеннями параметрів". – Зінченко. – 2021. – Харків: Харківський національний університет радіоелектроніки.

УДК 004.08

***Коваль Юлія Володимирівна**, здобувач вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
***Науковий керівник: Писаренко Андрій Володимирович**, к.т.н.,*
доцент кафедри інформаційних систем та технологій
КПІ ім. Ігоря Сікорського, Україна

АНАЛІЗ ВЕЛИКИХ МАСИВІВ ТЕКСТОВОЇ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ КЛЮЧОВИХ СЛІВ ТА ФРАЗ ЗАСОБАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Анотація. Аналіз великих масивів текстової інформації з використанням ключових слів та фраз є дуже актуальним, особливо в наш час, коли кількість інформації, з якою ми стикаємося щодня, стає все більшою та складнішою для аналізу. Для розв'язання цього завдання застосовуються різноманітні методи та технології, включаючи машинне навчання, глибоке навчання та інші.
КЛЮЧОВІ СЛОВА: АНАЛІЗ ТЕКСТОВОЇ ІНФОРМАЦІЇ, КЛЮЧОВІ СЛОВА, АНАЛІЗ ТЕКСТУ, МАШИННЕ НАВЧАННЯ, ГЛИБИННЕ НАВЧАННЯ.

Abstract. Analyzing large amounts of textual information using keywords and phrases is very relevant, especially nowadays, when the amount of information we encounter every day is getting bigger and more difficult to analyze. Various methods and technologies, including machine learning, deep learning, and others, are used to solve this problem.
KEY WORDS: TEXT INFORMATION ANALYSIS, KEYWORDS, TEXT ANALYSIS, MACHINE LEARNING, DEEP LEARNING.

Вступ. В сучасному світі кількість текстової інформації зростає експоненційно, і з кожним днем її обсяг стає все більшим та більшим. Це свою чергу створює величезне завдання для багатьох організацій та підприємств, які мають аналізувати значні обсяги тексту, отримання корисної інформації та приймати на основі цього важливі рішення. У таких умовах аналіз великих масивів текстової інформації з використанням ключових слів та фраз стає дедалі більш актуальним та важливим завданням. Ця робота присвячена дослідженню та покращенню процесу аналізу великих масивів текстової інформації з використанням ключових слів та фраз, а також процесу підготовки даних до нього. Основними методами та алгоритмами аналізу тексту є переробка даних, обробка природної мови, Байесова класифікація, кластеризація, машинне навчання, метод векторного представлення слів та глибинне навчання.

Основна частина. Аналіз тексту дозволяє швидко отримувати точну інформацію. Цей процес включає в себе підготовку текстових даних, аналіз, візуалізацію отриманої інформації, трактування результатів та формулювання висновків. Загалом, аналіз тексту з використанням ключових слів та фраз дозволяє здійснювати ряд важливих завдань, наприклад таких як, розуміння стану справ у певній галузі, виявлення попиту на товари та послуги, аналіз розпізнаного тексту з голосових повідомлень, виявлення схожих документів, покращення обробки та індексації текстової інформації, виявлення негативних відгуків та багато іншого.

Підготовка текстових даних перед аналізом є дуже важливим етапом і може суттєво впливати на результати аналізу. Текстові дані, як і інші джерела інформації включають в себе велику кількість зайвого, непотрібного та навіть шкідливого контенту. Перед початком аналізу текстових даних необхідно провести підготовку, яка включає в себе очищення даних від зайвих символів,

приведення до нижнього регістру, токенізацію, лематизацію та видалення стоп-слів. Якщо не здійснювати підготовку текстових даних, то результати аналізу можуть бути неточними та неправильними. Наприклад, якщо не видаляти стоп-слова, то аналіз може видавати високу важливість таких слів як «і», «або», «та», які не несуть суттєвої інформації для аналізу. Важливо зазначити, що дані можуть бути зібрані з різних ресурсів та мати купи помилок і неточностей. Тому перед тим, як проводити аналіз даних, необхідно виконати їх підготовку. Попередня обробка масиву текстових даних – це важлива складова будь-якого проекту, пов'язаного з аналізом тексту. Доцільність використання попередньої обробки текстових даних перед їх подальшим аналізом залежить від конкретної задачі. Однак, взагалі можна сказати, що вона допомагає покращити якість та ефективність обробки тексту. Видалення службових тегів та зайвих символів дозволяє отримати чистий текст, що є основою подальшого аналізу. Розділення тексту на токени допомагає відокремити окремі слова та фрази, що необхідно для проведення подальшого аналізу. Приведення тексту до нижнього регістру дозволяє уникнути проблем з узгодженням словникових форм слів. Видалення стоп-слів дозволяє уникнути включення найбільш часто зустрічаються слів, що не несуть значущої інформації для аналізу. Лематизація дозволяє зменшити кількість унікальних слів у тексті та врахувати форму слова, що може бути важливим для точної обробки тексту. Векторизація дозволяє перетворити текст у числовий вектор, що може бути використано для подальшого машинного навчання. Застосування попередньої обробки даних допомагає покращити якість моделей машинного навчання, знижує розмірність простору векторів, що зменшує вимоги до обчислювальних ресурсів та забезпечує більш точне інтерпретування результатів. Окрім того, цей підхід дозволяє покращити швидкість роботи алгоритмів, що має особливу важливість в задачах реального часу, таких як аналіз соціальних медіа, класифікація повідомлень тощо [1]. Алгоритм попередньої обробки великих масивів текстової інформації, що дозволяє належним чином підготувати дані до аналізу, складається з семи етапів: видалення службових елементів, видалення символів, токенізація, приведення до нижнього регістру, видалення стоп-слів, лематизація, векторизація. Таким чином, попередня обробка даних є важливим етапом у побудові моделей машинного навчання, оскільки вона допомагає покращити якість та швидкість роботи алгоритмів, знизити розмірність простору векторів та забезпечити більш точне інтерпретування результатів.

Наступним етапом в аналізі великих масивів текстової інформації з використанням ключових слів та фраз є обробка підготовлених даних. Попередньо оброблений та векторизований текст є ключовим елементом в аналізі текстових даних, оскільки це дозволяє відокремити важливу інформацію від надлишкової та зробити її придатною для подальшого аналізу. Крім того, такий текст можна класифікувати за певними критеріями, що дозволяє здійснювати різноманітні завдання. Методи аналізу векторизованого тексту є потужним інструментом для отримання нових знань та розуміння великих масивів текстової інформації. Вони дозволяють здійснювати швидкий та ефективний аналіз текстів та знаходити корисну інформацію, що може допомогти приймати рішення в різних областях діяльності. Для аналізу великих масивів текстової інформації було використано Наївний Баєсовий класифікатор [2]. Ця класифікація є потужним методом для аналізу великих масивів текстової інформації з використанням ключових слів та фраз. Вона може бути використана для класифікації текстів за тематикою, розпізнавання образів, визначення тональності текстів або для виявлення спаму. Наївний Байєсівський класифікатор приймає рішення про класифікацію на основі максимальної умовної ймовірності: він призначає вхідний зразок до класу, для якого умовна ймовірність максимальна. Тобто, алгоритм визначає ймовірність належності нового зразка до кожного класу, на основі розрахованих умовних ймовірностей, і призначає зразок до класу з найвищою ймовірністю [3]. Для прискорення аналізу великих масивів текстової інформації без

втрати точності, або з її підвищенням було проведено експериментальні дослідження, де наївному Байєсівському класифікатору надавався для аналізу певний відсоток слів з тексту (слова були взяті з векторизованого масиву, який був сформований в попередній обробці тексту). Результати дослідження, представлені на рисунках 1, 2, показали, що кращім відсотком для аналізу є проміжок 9%-21% слів тексту. Було проаналізовано тексти, що включать в себе десятки тисяч слів. З використанням зазначеного проміжку слів у аналізі, текст аналізується швидше та більш точно на відміну від більшої кількості, що можна побачити на гістограмі 1 зі значенням з 10%, 60% та 100%. Відносний час аналізу збільшився на 46% та 91% відносно 10% обробленого тексту, а точність зменшилась на 7% та 3%, відповідно. Забагато інформації, що надається класифікатору, починає вводити його в оману, через те, що в тексті можуть бути присутні рідковживані слова, які мають високу вагу, бо їх вірогідність появи дуже низька, але вони заважають якісно оцінити та проаналізувати текст. Загально прийнято проводити аналіз на основі 100% тексту, і що нетреба нехтувати жодним словом, але якщо існує потреба проаналізувати великі масиви текстових даних швидше, то це можна зробити описаним чином.

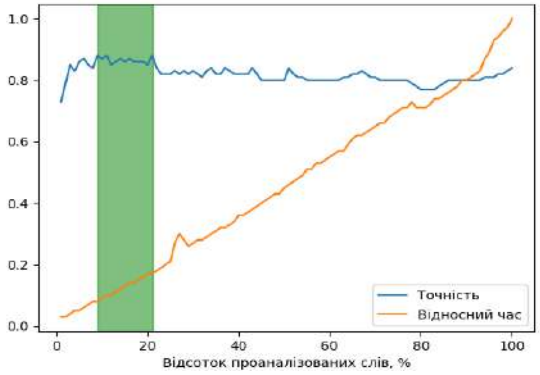


Рисунок 1 – Загальний графік показників якості в залежності від відсотку проаналізованих слів

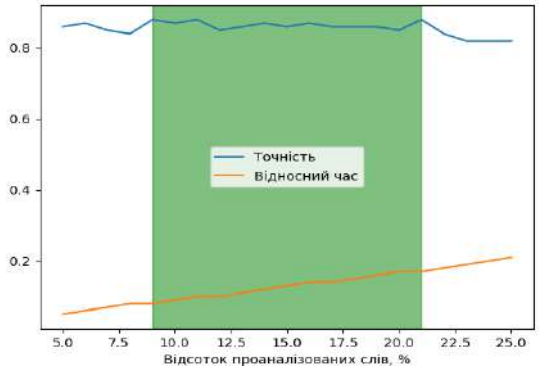
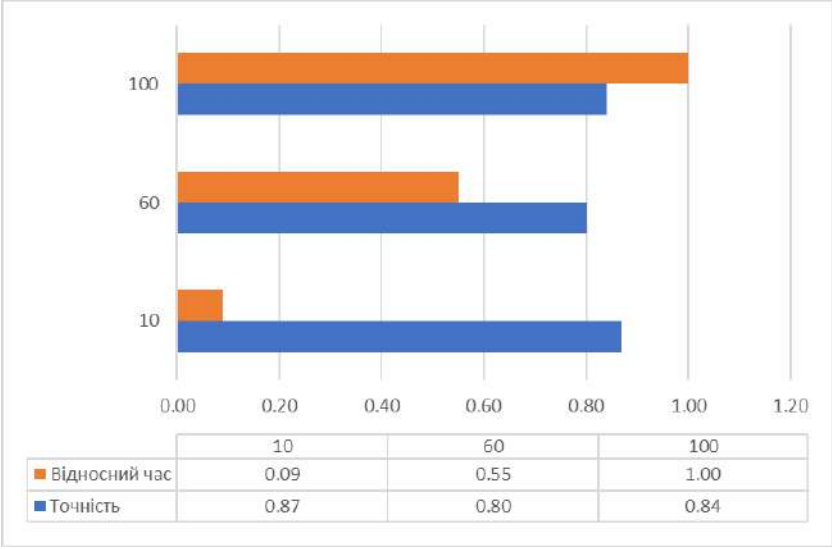


Рисунок 2 – Графік показників якості в залежності від відсотку проаналізованих слів у проміжку 5%-25%



Гістограма 1 – Гістограма показників якості класифікації великих масивів текстових даних в залежності від відсотку проаналізованих слів

Наступним етапом є візуалізація даних. Це процес графічного представлення інформації з метою зрозуміти та знайти корисну інформацію. Після отримання результатів

аналізу векторизованого тексту, візуалізація може допомогти в інтерпретації даних та виявленні нових закономірностей. Наприклад, графіки, діаграми, хмари слів, дерева вирішень та інші візуальні інструменти можуть бути використані для відображення різних аспектів даних, таких як кількість згадувань певної теми, зв'язки між словами, емоційний відтінок тексту та інші. Візуалізація може допомогти зрозуміти складні зв'язки між даними, виявити потенційні проблеми та відкрити нові можливості [4]. Отже, візуалізація даних може бути дуже корисним інструментом наступним етапом після аналізу векторизованого тексту для отримання нових знань та розуміння даних.

На останньому етапі аналізу великих масивів текстової інформації проводиться трактування результатів та формулюються висновки. Наприклад, можна зробити висновки про те, які ключові слова та фрази використовуються в тексті, яка тема є основною, та які питання стосуються дослідження. Ці висновки можуть бути корисні для прийняття рішень та розробки стратегій відповідно до результатів аналізу.

Висновки. На сьогоднішній день аналіз великих масивів текстової інформації з використанням ключових слів та фраз є розповсюдженим та актуальним. Перед початком аналізу потрібно провести попередню підготовку та обробку масиву текстових даних, у підготовці є доцільним наявність наступних етапів: видалення службових елементів, видалення символів, токенізація, приведення до нижнього регістру, видалення стоп-слів, лематизація, векторизація. Для аналізу було використано Наївний Баєсовий класифікатор. Задля підвищення показників якості цього процесу пропонується аналізувати певний відсоток слів тексту, що представляються у вигляді векторизованого масиву, а саме краще підходить використання проміжку 9%-21% слів тексту. Таким чином зменшується час аналізу та підвищується його точність. Область аналізу текстових даних є досить активною галуззю досліджень і розвитку, оскільки є багато різноманітних сфер застосування. У майбутньому можна очікувати подальшого розвитку цієї області, наприклад, може бути досліджене використання вузькоспеціалізованих слів, використання не тільки популярних слів, а й більш рідковживаних, використання довгих слів та абревіатур та інше.

Список інформаційних джерел

1. Аналіз впливу попередньої обробки текст / Гущин І.В., Сич Д.О. // Молодий вчений. – 2018. – № 10 (62). – С. 264-266.
2. Байесовский классификатор (Bayesian classifier). URL: https://wiki.loginom.ru/articles/bayesian_classifier.html.
3. Naïve Bayes Classifier Algorithm. URL: https://www.javatpoint.com/machine-learning-naive-bayes-classifier#:~:text=Na%C3%AFve%20Bayes%20Classifier%20is%20one,the%20probability%20of%20an%20object.
4. Технології візуалізації у світових дослідженняХ / Тютюнник А.В. // Open educational e-environment of modern University. – 2020. – № 9. – С. 161-168.

УДК 004.414

Бурятов Олексій Олексійович, *здобувач вищої освіти КПІ ім. Ігоря Сікорського, Україна*
Науковий керівник: Вітковська Ірина Іванівна, *старший викладач кафедри інформатики та програмної інженерії, КПІ ім. Ігоря Сікорського, Україна*

ПОБУДОВА ІНФРАСТРУКТУРИ ДЛЯ ВИСОКОПРОДУКТИВНИХ СИСТЕМ ТА ВПРОВАДЖЕННЯ ТЕНДЕНЦІЙ З ВИКОРИСТАННЯ BIG DATA ІНСТРУМЕНТІВ

INFRASTRUCTURE DEVELOPMENT FOR HIGH-LOAD SYSTEMS AND IMPLEMENTATION TRENDS OF USING BIG DATA TOOLS

Анотація. Діджиталізація процесів у повсякденному житті людини призводить до зростання цифрових носіїв інформації якими вона користується. Це призводить до необхідності обробки даних, які генерується у великих об’ємах щосекундно. Для обробки надвеликих масивів даних необхідні високі потужності, які зуміють виконати поставлену задачу. Попередні інструменти, якими користувалися інженери, стають неефективними в порівнянні з сьогоденними тенденціями. Нові методи та підходи допоможуть справитися з будь-якою задачею та знизити ризики втрати даних або недоступність сервісів. Інструменти по впровадженню та підтримці програмного забезпечення різноманітний, і особливо важливо правильно визначити доречність використання в тій чи іншій задачі.
КЛЮЧОВІ СЛОВА: Big Data, високонавантажені системи, автоматизація, Kubernetes, відмовостійкість.

Abstract. Digitalization of processes in everyday human’s life leads to increasing of usage smart devices, which they use. It leads to necessity of data processing, which generates in large amounts every second. In order to process enormous arrays of data, it is necessary to have high power, which will be able to perform the task. Previous tools, which used by engineers, become ineffective compared to current trends. New methods and approaches will help to cope with any task and reduce the risk of data loss or unavailability of services. Tools for implementing and maintaining software are diverse and it is especially important to correctly determine the appropriateness of use in a particular tasks.
KEY WORDS: Big Data, high-load systems, automation, Kubernetes, high availability.

Вступ. За статистикою, Big Data є однією з найбільш зростаючих галузей інформаційних технологій. Обсяг зберігання та отримання даних, згідно зі статистикою, збільшується в два рази кожні 1,2 роки. Наприклад, між 2012 та 2014 роками, обсяг даних, що щомісяця передаються мобільними мережами, збільшився на 81% [1, с. 195]. У 2014 році, обсяг мобільного трафіку склав 2,5 ексабайта на місяць, а вже у 2019 році - 24,3 ексабайта, за даними Cisco. Застосування технологій Big Data дуже широке. За результатами опитування IBM Institute, 53% використання Big Data стосується сфери клієнтського сервісу, 40% - операційної ефективності, а 7% - управління ризиками.

Таким чином, Big Data – це прогресуюча сфера технологій, яка набула поширення у багатьох сферах бізнесу і відіграє важливу роль у розвитку компаній.

Основна частина. Big Data (з. англ. – великі дані) – надвеликі масиви даних, які обробляються спеціальними програмними та технічними засобами, а також потребують високопродуктивного обладнання для швидкого та безперебійного аналізу даних [2, с. 12]. Зазвичай Big Data рішення потребують великі бізнеси, які збирають статистику зі

своїх продуктів, щоб у подальшому покращити сервіси надання послуг, використовуючи аналітику.

Розглядаючи можливі варіанти побудови потужного сховища та системи збору й обробки інформації, один із основних принципів – це створення потужного кластеру. На сьогоднішній день є купа рішень програмного забезпечення для Big Data продуктів – як безкоштовних, так і комерційних. Якщо обирати із open-source – це створення кластеру Kubernetes, Apache NiFi, Apache Spark, Confluent Kafka. Якщо обирати із комерційних пропозицій – це Cloudera Data Platform, Teradata тощо. Кожен сервіс розроблявся під окремі задачі, тому побудова універсального кластеру неможлива.

Однією із найпопулярніших платформ для управління ПЗ – це Kubernetes [3, с.1]. Головними перевагами використання сервісу є:

1. High availability – високий поріг відмовостійкості у разі виходу з ладу фізичного серверу, пошкодження обладнання, недоступність мережі тощо.
2. Open source – можливість використовувати безкоштовно, що є вагомим чинником для сучасного бізнесу, а також постійне вдосконалення користувачами цього сервісу, які зацікавленні в покращенні та усуненні багів;
3. Ізоляція ресурсів – кожне ПЗ запускається в окремому «віртуальному» середовищі, яке нічого не знає про інші застосунки;
4. Можливість розгортання в хмарі – багато провайдерів хмарних послуг (такі як Microsoft Azure, Amazon Web Services, Google Cloud Platform) інтегрували Kubernetes до свого сховища і тому є можливість використання цього сервісу в хмарі.

З чого починати при створенні високопродуктивних систем для обчислення надвеликих масивів даних? Перш за все необхідно вирішити питання з наявними ресурсами, а саме: обрати фізичні сервери (або хмарного провайдера сервісів) та їх потужність, визначити пропускну здатність мережевого каналу, підрахувати об’єми та швидкість дискового простору.

Автоматизація – це необхідна частина розробки, впровадження, оновлення та інтегрування застосунків і сервісів. Працюючі в команді, необхідно максимально автоматизувати процеси, адже кожен учасник розробки повинен розуміти як працює ціла система та можливість легко й швидко працювати разом. Для цього існують декілька інструментів, які допоможуть автоматизувати процеси від написання коду, зборки образів та розгортання застосунків, а також для створення самої інфраструктури як на наземному кластері, так і в хмарі:

1. Terraform – програмне забезпечення, яке використовується для створення, редагування, видалення та версіонування хмарної інфраструктури. Цей інструмент передбачає більше ніж 200 хмарних провайдерів та наземної інфраструктури [4, с. 5];
2. Ansible – інструмент по автоматизації виконання коду для великих кластерів фізичних серверів. За допомогою скриптів, або як ще називають – playbooks (з англ. – сценарій) можна одночасно виконувати поставлені задачі, зберігаючи час та консистентість інфраструктури [4, с. 6];
3. CI/CD – (continuous integration and continuous deployment – з англ. – неперервна інтеграція та неперервне розгортання) – практика, яка автоматизує процес тестування та розгортання програмного забезпечення під час написання коду. Цей інструмент важливий тим, що він реагує на зміни в коді, написані розробником, а також має логування й версіонування, адже при неуспішних тестах інфраструктура зберігає свій попередній працюючий стан. Такі інструменти є в GitLab, GitHub, Jenkins, TeamCity тощо.

Якщо розглядати вищеперераховані інструменти у площині високопродуктивних систем та обчислень, то вони просто необхідні – в інакшому випадку є ймовірність недоступності застосунків, що може вкрай негативно вплинути на роботу усіх налаштованих процесів, адже часто один процес залежить від іншого.

Висновки. Отже, підсумовуючи вищесказане, можна прийти до висновку, що Big Data – це один із головних напрямів сучасного бізнесу, який має великі масиви даних для обробки. Для створення потужних обчислень великих масивів даних, необхідно потурбуватися про високонавантажену систему та сервіси, які будуть займатися обробкою. Одним із основних інструментів є Kubernetes, який оркеструє запущені сервіси, ізолюючи один від одного, контролює використання ресурсів під кожен процес та гарантує високу доступність навіть при збоїв системи. Усім цим необхідно керувати інструментами автоматизації, які пришвидшують планові та позапланові оновлення, логують дії та роблять їх консистентими.

Список інформаційних джерел

1. Глоба Л.С., Курдеча В.В., Дундяк Р.Р. , РОЗВАНТАЖЕННЯ МОБІЛЬНИХ МЕРЕЖ ЗА ДОПОМОГОЮ ПОЄДНАННЯ ТЕХНОЛОГІЙ WI-FI OFFLOAD ТА LTE. ХЕНДОВЕР МІЖ ДВОМА ТЕХНОЛОГІЯМИ [Електронний ресурс] / Національний технічний університет України «Київський політехнічний інститут» – Режим доступу до ресурсу: http://molodyvcheny.in.ua/files/journal/2016/6/48.pdf
2. В. Г. Саріогло. “Великі дані” як джерело інформації' та інструментарій для офіційної статистики: потенціал, проблеми, перспективи [Електронний ресурс] / В. Г. Саріогло // ISSN 2519-1853 СТАТИСТИКА УКРАЇНИ. – 2016. – Режим доступу до ресурсу: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=su_2016_4_5
3. Eunsook Kim; Kyungwoon Lee; Chuck Yoo. On the Resource Management of Kubernetes [Електронний ресурс] / Eunsook Kim; Kyungwoon Lee; Chuck Yoo // IEEE. – 2021. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/9333977.
4. Single and Multi-Cloud Disaster Recovery Management using Terraform and Ansible [Електронний ресурс] // National College of Ireland Project Submission Sheet School of Computing. – 2019. – Режим доступу до ресурсу: https://norma.ncirl.ie/4144/1/rambaraththiyagarajan.pdf

УДК 004.414

Чеботарьова Анна Владиславівна, *здобувач вищої освіти Дніпровського національного університету імені Олеся Гончара, Україна*
Науковий керівник: Сидорова Марина Геннадіївна, *кандидат технічних наук, доцент кафедри математичного забезпечення ЕОМ Дніпровського національного університету імені Олеся Гончара, Україна*

ГЕНЕРАЦІЯ МУЗИКИ З УРАХУВАННЯМ ЕМОЦІЙНОЇ СКЛАДОВОЇ КОРИСТУВАЦЬКОГО ЗАПИТУ

Анотація. Штучний інтелект посідає важливе місце в житті людей та бізнесі. З його допомогою можна значно пришвидшити процеси та розробити економічно вигідні рішення. Все більше компаній використовують згенеровану музику замість музики, створеною людиною, оскільки це швидкий та дешевий підхід. Велика кількість генераторів музики пропонує користувачу обрати багато музичних параметрів для генерація музики, вплив яких розуміє тільки професійний музикант. Представлений підхід пропонує генерувати музику на основі емоції, яку користувач хоче почути у музиці. Це дозволяє зробити процес генерації музики більш простішим та швидшим для користувачів, які не мають музичної освіти або музичного досвіду.
КЛЮЧОВІ СЛОВА: Нейронні мережі, Генеративні змагальні нейронні мережі (GAN), Класифікація тексту за емоційною складовою, NLP.

Abstract. Artificial Intelligence occupies an important place in people's lives and business. With AI help people can significantly speed up processes and develop cost-effective solutions. More companies are using generated music instead of human-made music because it is a quick and cheaper approach. Most music generators offer the user to choose many musical parameters, the effect of which is understood only by a professional musician. The presented approach proposes to generate music based on the emotion that the user wants to hear in the generated music. This makes the process of music generation easier and faster for users with no musical education or musical experience.
KEY WORDS: Neural networks, Generative adversarial neural networks (GAN), Text classification by emotions, NLP.

Вступ. Генерація видів мистецтва, таких як музика, зображення, текст – є однією з найскладніших завдань штучного інтелекту. Багато існуючих архітектур можуть тільки частково згенерувати музику, зображення чи текст, які б містили такі складні для математичних алгоритмів, але прості для людського розуміння речі як емоцію, підтекст, порівняння, тощо. Найуспішнішою серед таких моделей є генеративні змагальні нейронні мережі (GAN) [1]. На відміну від інших моделей штучного інтелекту, вони можуть генерувати складні в машинному розумінні речі, що призвело до їх широкого використанні State-of-the-art [2] напрямку. Через емоції людина може описати багато складних та незрозумілих речей, в своєму роді емоції є інтуїтивним засобом комунікації між людьми, але комунікація між людьми та машинами все ще залишається дуже складною в деяких аспектах. Одним з таких аспектів є генерація музики через програми-генератори, де користувачу потрібно обрати багато комплексних параметрів, що робить генерацію музики дуже складним процесом з людської точки зору. Представлений проект пропонує спростити генерацію музики шляхом заміни параметрів генерації на описове речення, яке містить емоцію [3], яку користувач хоче почути у згенерованій музиці. Такий підхід підійде користувачам, яким потрібно викликати згенерованою музикою емоції у людей, збільшивши їх досвід від прослуховування. Наприклад, це можуть використовувати у розробці ігор, де в конкретний момент часу потрібно викликати у гравця необхідну емоцію, в презентаціях

художніх виставок, театральних виставах, як музичний супровід у відео. Використання такого підходу є економічно вигіднішим для генерації нескладної, з музичної точки зору мелодії, оскільки не потребує наймати музикантів та ансамблі.

Основна частина. У цій роботі розроблено програмне забезпечення, що вирішує описану вище задачу та з двох частин. Перша частина – комплекс підходів для класифікації користувацького запиту за класом емоції. Друга частина – генерація музики у вигляді спектрограм та подальше перетворення згенерованих спектрограм у музичну послідовність формату .wav.

Загальний алгоритм полягає у тому, що користувач вводить речення (запит), що класифікується за емоцією (любов, злість, здивування, радість, страх, смуток) та співвідноситься з жанром музики, за яким і буде здійснюватися генерування нової композиції (рис.1.).

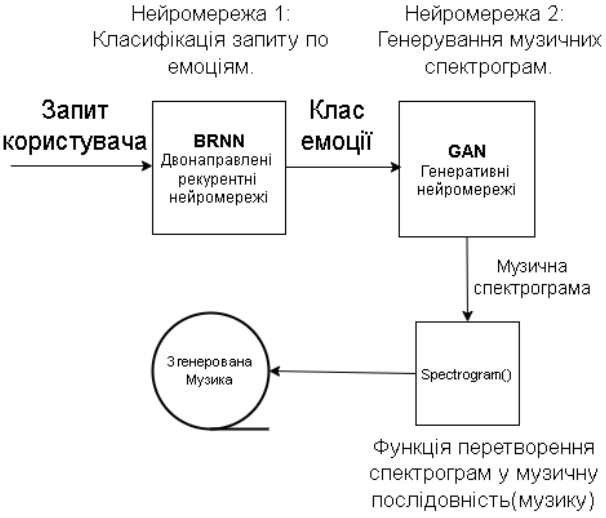


Рис.1. Загальна схема

Причина, по якій для кожного жанру музики розроблена та навчена окрема нейромережа полягає в тому, що нейромережі типу GAN є дуже складними у навчанні та розробці, оскільки потребують великих потужностей та довго навчаються. Розробка GAN нейромережі для генерації всіх жанрів музики, яка навчалася на всіх жанрах музики значно ускладнює архітектуру, потребує більше потужностей та часу на навчання.

Класифікація тексту є частиною NLP (Natural Language Processing) області, яку вважають областю штучного інтелекту через її складність. Для задачі класифікації тексту [4] було обрано класифікатор на основі архітектури двонаправленої рекурентної нейронної мережі (BRNN) з “ефектом пам’яті” – LSTM, оскільки рекурентні нейромережі створені для роботи з послідовністю, двонаправленість забезпечує обробку слів в прямому та зворотньому напрямках, що збільшує якість передбачень, а Long short-term memory (LSTM) дозволяє розуміти зміст більш довгих послідовностей. Класифікатор навчався на наборі даних, розділених на текстові дані (твіти користувачів мережі Twitter) [5], які належать одному з шести класів емоцій.

Перед початком тренування класифікатора, дані були очищені від повторюваних та стоп-слів розділені на тренувальний (70%) та валідаційний (30%) набори, токенізовані за допомогою токенайзера та пропущені через Glove Embedding. Щоб досягти точність у 96% класифікатору необхідно близько 10 хвилин. Дані для перевірки точності класифікації були обрані з книги “Гобіт, або Туди і Звідти”, автора Дж.Р.Р. Толкіна,оскільки вони є складнішими за інтерпретацією, ніж тренувальні дані, результат показано на рис.2 (Повний текст: *”Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort.”*)

```
Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell
1/1 [==============================] - 1s 604ms/step
1/1 [==============================] - 1s 844ms/step
joy : 0.9873606562614441
```

Рис.2. Результат класифікації

Друга частина є складнішою у побудові та навчанні. За основу взятий підхід, запропонований Google AI на конференції ICLR 2019 [6]. Він полягає в тому, щоб вчити нейромережу з GAN архітектурою генерувати не саму музичну послідовність, а музичні спектрограми, оскільки GAN-и показують найкращий результат на задачах генерації зображень. Архітектура GAN складається з генератора, чия задача генерувати максимально якісні зображення (fake samples) та дискримінатора, чия задача повертати до генератора відсоток “вдалих”

зображень – це зображення, які дискримінатор класифікував як реальні (real samples), хоча вони згенеровані генератором. Завдяки такій архітектурі генератор та дискримінатор "навчають" один одного. Для цього проєкту генератор та дискримінатор нейромережі мають згорткову архітектуру, оскільки математична операція згортки застосовується для роботи з зображеннями. На відміну від інших нейромереж, де задачею є мінімізація функції втрат, задачею GAN нейромережі є стабілізація функції втрат та максимізація функції точності. Досягнення такої стабільності є найскладнішою частиною розробки нейромереж такого типу, оскільки стабілізація функцій є складним математичним процесом. На основі публікації [7] вдалося частково стабілізувати GAN нейромережу (рис.3).

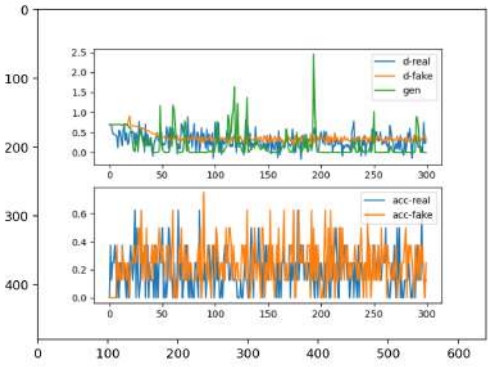


Рис 3. Графіки функцій втрат та точності

Наведена нейромережа тренувалася на спектрограмах, створених з 30-секундних композицій жанру блюз. Для забезпечення стабільності було застосовано кілька підходів: розділене навчання дискримінатора: спочатку на даних реальних спектрограмах, після навчання на згенерованих спектрограмах, заміна міток класів реальних та згенерованих зображеннях: замість міток 0 (згенеровані зображення) та 1 (реальні зображення) використано функцію генерації рандомних чисел Гаусівського розподілу – межі генерації для згенерованих зображень: 0 - 0.2, для реальних зображень: 0.7 - 1.1., заміна функцій Sparse градієнту (MaxPool, ReLu) на Conv2dTranspose та LeakyReLu відповідно.

Результат навчання на 50 епохах (1.5 години, 10 батчів по 10 об'єктів) наведено на рис.4.

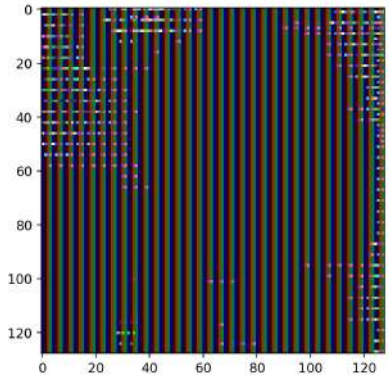


Рис.4. Згенерована спектрограм після навчання протягом 50 епох

Без застосування цих підходів нейромережа замість спектрограм буде генерувати шум (рис.5), функція втрат у дискримінатора після 2 епохи буде близька до 0, тоді як функція втрат генератора буде постійно зростати (рис.6.)

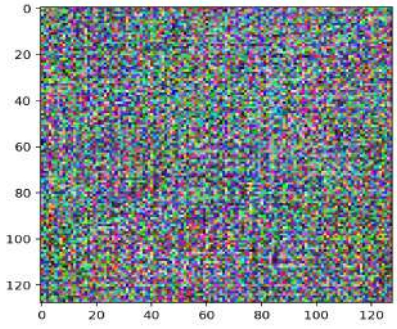


Рис 5. Згенероване шумове зображення.

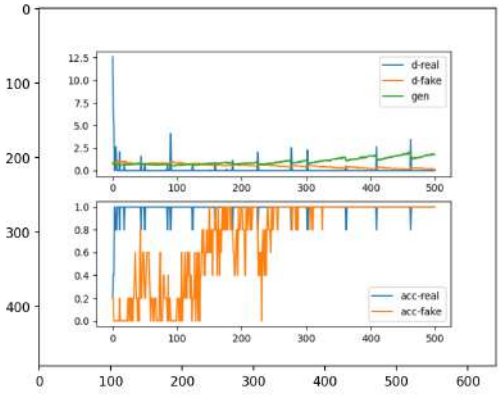


Рис.6. Графіки функцій втрат та точності

Висновки. Розроблено програмне забезпечення на основі архітектур нейронних мереж для генерування музики з врахування емоційної складової та розглянуто підходи для оптимізацій та покращення якості генеративних змагальних моделей нейронних мереж .

Список інформаційних джерел

1. Generative Adversarial Networks / Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio // Advances in Neural Information Processing Systems 27 (NIPS 2014). – 2014. – Vol. 27. – P. 2672–2680
2. Nature Inspired Computation and Swarm Intelligence / Xin-SheYang // Academic Press. – 2020. – 442 p.
3. Music Emotion Recognition: Toward new, robust standards in personalized and context-sensitive application / J.S.Gómez-Cañón; E.Cano,T.Eerola; P.Herrera; Xiao Hu; Yi-Hsuan Yang // IEEE Signal Processing Magazine, Vol. 38 (6), 2021, P. 106-114
4. Review of Text Classification in Deep Learning / Qi Wang, Wenling Li, Zhezhi Jin // Open Access Library Journal. – Vol. 8 (03). – 2021. – P. 1-8. 5. Emotions dataset for NLP. URL:https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp
6. GANSYNTH: ADVERSARIAL NEURAL AUDIO SYNTHESIS / J.Engel, K.K.Agrawal, S.Chen, I.Gulrajani, C.Donahue, A.Roberts // ICLR. – 2019. URL: https://arxiv.org/abs/1902.08710
7. How to Train a GAN? Tips and tricks to make GANs work / Soumith Chintala, Emily Denton, Martin Arjovsky, Michael Mathieu // NIPS 2016. URL: https://github.com/soumith/ganhacks

УДК 004.92

Трофимов Данило, *здобувач вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник: Родіонов Павло Юрійович,
доцент кафедри інформатики та програмної інженерії
КПІ ім. Ігоря Сікорського, Україна

ОСОБЛИВОСТІ СТВОРЕННЯ ШЕЙДЕРНИХ ПРОГРАМ ДЛЯ РЕАЛІЗАЦІЇ ОСВІТЛЕННЯ

Анотація. У роботі розглядаються теоретичні засади та практичні аспекти реалізації моделі освітлення Фонга. Запропоновано підхід до реалізації моделі освітлення на базі графічного прикладного інтерфейсу WebGL. Проаналізовано особливості реалізації даної моделі освітлення та запропоновано відповідні шейдерні програми.
КЛЮЧОВІ СЛОВА: модель освітлення, фрагментний шейдер, вершинний шейдер.

Abstract. The paper examines the theoretical foundations and practical aspects of the implementation of the Fong lighting model. An approach to the implementation of the Fong lighting model based on the WebGL graphical application interface is proposed. The peculiarities of the implementation of this lighting model were analyzed from the point of view of the load on the hardware of the information system.
KEY WORDS: fragment shader, vertex shader, lighting model.

Вступ. Розповсюдженою проблемою у сфері комп'ютерної графіки традиційно вважається реалізація освітлення. На даний момент існує множина моделей освітлення, кожна з яких має власні переваги та недоліки. Проте важливою науковою задачею видається систематизувати та узагальнити теоретичні засади реалізації освітлення, а також дослідити методи реалізації конкретної моделі освітлення на базі графічного прикладного інтерфейсу WebGL.

Основна частина. Процеси моделювання освітлення базуються на законах геометричної оптики, таких як закони заломлення, відбиття, прямолінійності тощо. Енергія світла, що падає на поверхню від джерела світла, може бути частково поглинута, відбита або пропущена через поверхню. Кількість енергії, що поглинається, відбивається і пропускається залежить від довжини λ світлової хвилі. Об’єкт можна побачити тільки тоді, коли він відбиває або пропускає світло. Властивості відбитого світла, які визначають колір поверхні, залежать від форми і напрямку джерела світла, від орієнтації поверхні, на яку падає світло та від властивостей самої поверхні. Цей факт враховується в різних моделях освітлення. Дійсна теорія світла є досить складною з теоретичної та обчислювальної точок зору, тому будуються спрощені моделі освітлення [1].

Модель Фонга аналогічна до моделі Гуро, але, на відміну від нього, тут інтерполюються не значення інтенсивності відбитого світла, а значення вектора зовнішньої нормалі. Тобто, модель Фонга полягає в побудові для кожного пікселя вектора, що відіграє роль нормалі і цей вектор використовується для обчислення інтенсивностей компонент відбитого світла в кожній точці згідно з вибраною моделлю освітлення. Тому зафарбовування на основі

моделі Фонга вимагає значно більше обчислень, оскільки інтерполюються три векторних компоненти, проте одержуються більш реальні графічні зображення [1].

Реалізація освітлення на базі API WebGL передбачає написання вершинних та фрагментних шейдерів. Нижче наведено лістинг вершинного шейдера WebGL-програми, у якій реалізовано модель освітлення.

```
void main() {
gl_Position = uProjectionMatrix * uModelViewMatrix *
vec4(aVertexPosition, 1.0);
vColor = aVertexColor;
vNormal = normalize(vec3(uNormalMatrix *
vec4(aNormal, 0.0)));
vViewDirection = normalize(-vec3(uModelViewMatrix *
vec4(aVertexPosition, 1.0)));
}
`;
```

Нижче запропоновано лістинг фрагментного шейдера для реалізації моделі освітлення.

```
void main() {
vec3 ambient = uAmbientColor * vColor;
vec3 lightDirection = normalize(uLightPosition -
gl_FragCoord.xyz);
vec3 normal = normalize(vNormal);
vec3 diffuse = uDiffuseColor * vColor * uLightColor;
vec3 reflectionDirection = reflect(-lightDirection, normal);
vec3 viewDirection = normalize(vViewDirection);
float specularFactor = pow(max(dot(reflectionDirection,
viewDirection), 0.0), uShininess);
vec3 specular = uSpecularColor * specularFactor *
uLightColor;
vec3 finalColor = ambient + diffuse + specular ;
gl_FragColor = vec4(finalColor, 1.0);
}
`;
```

Оскільки поверхня має передню та задню грані, кожна сторона має свою нормаль. Наприклад, поверхня, перпендикулярна до осі z, має передню поверхню, яка звернена до позитивного напрямку осі z, і задню поверхню, яка звернена до негативного напрямку осі z. Їх нормалі дорівнюють (0, 0, 1) і (0, 0, –1) відповідно.

У 3D-графіці ці дві грані розрізняються порядком, у якому вказані вершини під час рендерингу поверхні. Коли відбувається рендеринг із зазначенням вершин, передня грань – це та, вершини якої розташовані за годинниковою стрілкою, якщо дивитися вздовж нормалі грані [2].

Висновки. У роботі було систематизовано теоретичні основи щодо реалізації освітлення при створенні графічних сцен. Було запропоновано підходи до написання шейдерних програм для реалізації моделі освітлення, а також наведено приклад таких шейдерів. В якості перспектив подальших наукових досліджень може бути запропоновано створення порівняльного аналізу реалізації моделі освітлення Фонга на базі різних графічних програмних інтерфейсів.

Список інформаційних джерел

1. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.
2. Matsuda K. & Lea R. (2013). Webgl programming guide : interactive 3d graphics programming with WebGL. Addison-Wesley.

УДК 004.414

***Митник Денис Олександрович**, здобувач вищої освіти КПІ ім. Ігоря Сікорського, Україна*
***Науковий керівник: Гавриленко Олена Валеріївна**, кандидат фізико-математичних наук, доцент, доцент кафедри інформаційних систем та технологій ФІОТ КПІ ім. Ігоря Сікорського, Україна*

ПОРІВНЯННЯ ЛАНЦЮГІВ МАРКОВА ТА НЕЙРОМЕРЕЖ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ГЕНЕРАЦІЇ ВІРШІВ

Анотація. В даній статті обумовлено важливість задачі генерації віршів. Описано алгоритми генерації віршів за допомогою ланцюгів Маркова та нейромереж. Охарактеризовано переваги та недоліки кожного з підходів та здійснено їх порівняння.
КЛЮЧОВІ СЛОВА: генерація віршів, ланцюги маркова, нейромережі, natural language processing.

Abstract. This article states the importance of the task of generating poems. Algorithms for generating poems using Markov chains and neural networks are described. The advantages and disadvantages of each of the approaches are characterized and their comparison is made.
KEYWORDS: verse generation, Markov chains, neural networks, natural language processing.

1. Вступ

Задача генерації тексту та віршів є однією з найбільш актуальних в сфері Natural Language Processing. Існує досить велика кількість підходів та моделей для генерації тексту, проте багато з них мають досить велику різницю в порівнянні зі справжньою людською мовою, отже постає задача контексту, тобто логічно звязаного змісту згенерованого тексту. Задача генерації віршів розкриває генерацію не лише з боку логічного змісту, а й ритму, рими та емоцій, розв'язання цієї задачі дозволяє зробити продукти які використовують згенеровану мову більш привабливими для користувачів, такі як голосові помічники або чат-боти. Також генерація віршів може бути одним зі способів генерації синтетичних даних для тренування мовних моделей. Однією з проблем які виникають в творчості є проблема авторського права і вона також може бути вирішена за допомогою генерації віршів. З теоретичної точки зору ця задача відкриває нові грані розуміння мов з боку computational linguistics, а саме пояснює яке формальне представлення мови та які методи є найбільш доречним для моделювання її комп'ютером.

2. Ланцюги Маркова

Ланцюг Маркова це ймовірнісна модель яка описує послідовність подій і ймовірність наступної події залежить лише від поточного стану моделі [1]. У випадку генерації віршів ланцюги Маркова можуть використовуватися для знаходження наступного слова за наявними вже у вірші.

На першому кроці алгоритму необхідно розділити тренувальний корпус на пари вікно (слова які передують) та слово яке слідує за цим вікном. В результаті ми отримуємо структуру даних наступного вигляду [2]:

Рисунок 1. Структура даних для збереження вікон та ймовірностей наступних слів

Обираємо початкову послідовність для генерації, після цього виділяємо з послідовності вікно зі слів, знаходимо ймовірні наступні слова та обираємо з ймовірностями наступне слово, даних алгоритм продовжується поки вірш не буде мати необхідну довжину. Наведемо псевдокод алгоритму:

ПОЧАТОК

1) Розбити корпус на пари вікно - наступне слово та розрахувати ймовірності наступного слова
2) Отримати початкову послідовність
3) Поки довжина згенерованої послідовності менша за очікувану:

a) Отримати вікно з послідовності
b) Обрати наступне слово за ймовірностями з корпусу
c) Додати наступне слово до згенерованої послідовності

КІНЕЦЬ

Наведений алгоритм є найпростішою реалізацією ланцюга Маркова для генерації віршів, проте даний алгоритм можна модифікувати аби він краще зберігав риму у вірші. Наприклад під час отримання наступного слова за вікном можна надавати більше перевагу слова, які римуються відповідно до обраної схеми римування, також в таком випадку необхідно обмежувати кількість слів у одному рядку вірша, аби розуміти яке слово останнє в рядку [3].

Перевагами даного алгоритму є швидкість імплементації і досить непогана імітація стилю вірша, проте простий статистичний підхід не завжди може зберегти ритм та риму вірша.

3. Нейромережевий підхід

Одним з основних напрямків NLP є методи на основі нейронних мереж. Задача генерації віршів такоє може вирішуватися цим підходом, однією з найбільш ефективних архітектур є рекурентні нейронні мережі.

Рекурентні нейронні мережі особливі тим, що в них звязки вузлів можуть бути з самими собою, що дозволяє даному типу мереж дуже якісно обробляти будь-які послідовності [4]. Однією з найефективніших реалізацій блоків рекурентної нейромережі є LSTM (Long Short Term Memory).

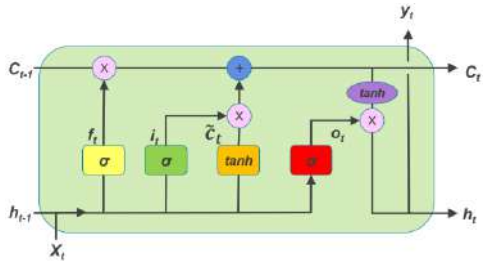


Рисунок 2. Структура LSTM блоку[5]

Цей блок особливий тим що він здатний вивчати довготривалі залежності, що в задачі генерації віршів є надзвичайно важливим для збереження ритму та змісту.

Блок LSTM складається з наступних частин:

– Input Gate
– Forget Gate
– Output Gate

Рівняння Gate:

$$i_t = \sigma\left(w_i\left[h_{t-1}, x_t\right] + b_i\right)$$

$$f_t = \sigma\left(w_f\left[h_{t-1}, x_t\right] + b_f\right)$$

$$o_t = \sigma\left(w_o\left[h_{t-1}, x_t\right] + b_o\right)$$

де

- i_t – *imprit gate*
- f_t – *forget gate*
- o_t – *output gate*
- σ – сигмоїд-функція
- w_x – ваги нейронів відповідного gate
- h_{t-1} – вихідні дані попереднього lstm блоку
- x_t – вхідні дані в момент часу t
- b_x – bias для відповідного gate

Рівняння для стану:

$$c_t\hat{} = tanh\left(w_c\left[h_{t-1}, x_t\right] + b_c\right)$$

$$c_t = f_t * c_{t-1} + i_t * c_i\hat{}$$

$$h_t = o_t * tanh\left(c_t\right)$$

Реалізувати генерацію віршів за допомогою lstm нейромережі можна наступним чином [6]:

ПОЧАТОК

1) Розбити тренувальний корпус на пари вікно-наступне слово
2) За допомогою даного датасету тренувати методом градієнтного спуску нейромережу
3) Зберегти ваги
4) Задати початкову послідовність
5) Поки довжина згенерованої послідовності менша за бажану:
   a) Отримати вікно з послідовності
   b) Подати вікно на вхід нейромережі та отримати наступне слово

c) Додати наступне слово до згенерованої послідовності

КІНЕЦЬ

Перевагами використання нейронних мереж для генерації віршів є їх більш складна лінійна трансформація параметрів, що дозволяє вивчити залежності в тексті та зберігати контекст [6]. Вони показують кращий результат та є дуже гнучкими щодо мотифікацій. серед недоліків цього підходу варто зазначити досить довгий час тренування та підбору гіпер параметрів моделі та досить нетривіальну реалізацію усіх компонентів моделі.

Висновки. Було описано такі алгоритми генерації віршів як ланцюги Маркова та нейромережі. В результаті порівняння алгоритмів було виявлено, що статистичні методи не завжди можуть вловити ритм та риму віршу, в той час як рекурентні нейронні мережі здатні якісно вловлювати контекст, ритм та риму віршу. Використання нейромереж для задачі генерації віршів при правильному підборі архітектури може досягати майже людських результатів.

Список інформаційних джерел

1. Manuel Brenner (2022). Understanding Markov Chains [Електронний ресурс] // Режим доступу https://towardsdatascience.com/understanding-markov-chains-cbc186d30649

2. Mehrab Jamee (2018). Making a Markov Chain Poem Generator in Python [Електронний ресурс] // Режим доступу:
https://medium.com/upperlinecode/making-a-markov-chain-poem-generator-in-python-4903d0586957

3. Gabriele Barbieri (2013). Markov constraints for generating texts with style [Електронний ресурс] // Режим доступу http://amsdottorato.unibo.it/5685/1/barbieri_gabriele_tesi.pdf

4. Alex Sherstinsky (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network [Електронний ресурс] // Режим доступу https://arxiv.org/pdf/1808.03314.pdf

5. Divyanshu Thakur (2018). LSTM and its equations [Електронний ресурс] // Режим доступу https://colah.github.io/posts/2015-08-Understanding-LSTMs/

6. Paul Mooney (2017). Poetry Generator (RNN Markov) [Електронний ресурс] // Режим доступу https://www.kaggle.com/code/paultimothymooney/poetry-generator-rnn-markov

УДК 004.8

Баран Данило Романович, *аспірант кафедри обчислювальної техніки,*
факультету інформатики та обчислювальної
КПІ ім. Ігоря Сікорського, Україна
Туганських Олександр Антонович, *здобувач вищої освіти,*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник: Писарчук Олексій Олександрович, *доктор технічних наук, професор,*
професор кафедри обчислювальної техніки, факультету інформатики та обчислювальної
КПІ ім. Ігоря Сікорського, Україна

МЕТОД АНАЛІТИЧНОГО ВИЗНАЧЕННЯ ПАРАМЕТРІВ НЕЛІНІЙНИХ МОДЕЛЕЙ ЗА СТАТИСТИЧНОЮ ВИБІРКОЮ ВИМІРІВ

Анотація. Запропоновано метод аналітичного визначення параметрів нелінійних моделей за статистичною вибіркою вимірів для задач Data Science. Метод базується на методології статистичного навчання в схемі диференціально-нетейлорівських перетворень. Наведено приклад застосування запропонованого методу з побудовою експоненційної моделі. Доведено ефективність запропонованого підходу, порівняно із класичними методами чисельних ітераційних розрахунків.
КЛЮЧОВІ СЛОВА: модель, статистичне навчання, Data Science, диференціальні перетворення.

Abstract. Analytical method of determining parameters for nonlinear models using statistical samples in Data Science is suggested here. It is based on statistical learning methodology in scheme of differential non-taylor transformations. As example of suggested method usage, the exponential model was built. The effectiveness of its application was confirmed using classical numerical iterative calculations as comparison.
KEY WORDS: model, statistical learning, Data Science, differential transformations.

Вступ. Сучасним прикладним задачам технологій Data Science притаманні нелінійність, значна динаміка зміни та надмірність статистичних вибірок досліджуваних процесів. При цьому висуваються жорсткі вимоги до оперативності, точності та достовірності результатів обробки, а також до інваріантності математичної моделі статистичного навчання до властивостей експериментальних даних. Це може бути досягнуто впровадженням аналітичних підходів до визначення параметрів нелінійних моделей. Традиційно, для вирішення цієї задачі використовуються чисельні методи апроксимації [1]. Проте вони мають значну обчислювальну складність та потребують апріорної інформації про межі зміни шуканих параметрів моделі. Це унеможливлює їх використання для задач сучасних задач Data Science. Тому ***актуальною*** є задача удосконалення методів статистичного навчання у напрямку подолання недоліків традиційних підходів. ***Метою*** досліджень є розробка методу аналітичного визначення параметрів нелінійних моделей за статистичною вибіркою вимірів

Основна частина. Суть методу аналітичного визначення параметрів нелінійних моделей за статистичною вибіркою – імплементація теоретико-емпіричного підходу до аналізу нелінійних процесів. Він базується на перенесенні якостей спрощеної моделі досліджуваного процесу на нелінійну, яка є більш адекватною до теоретичного опису процесу. Для отримання параметрів спрощеної моделі можуть пропонується використати будь-який метод статистичного навчання, наприклад метод найменших квадратів (МНК) Теоретичний опис математичної моделі може бути отриманий з аналітичних спостережень за природою вибірки даних. Значна, складність нелінійного теоретичного опису моделі унеможливлює застосування класичних статистичних алгоритмів згладжування. Запропонований метод полягає в багатоетапному отримання параметрів нелінійної моделі з використанням

диференційних перетворень [3] для мінімізації розходжень поліноміальної та нелінійної моделей. Такий підхід дозволяє поєднати відносну простоту поліноміальної моделі з більш прогностично стійкою нелінійною моделлю.

Вихідними даними для аналізу нелінійного процесу є дискретна вибірка параметрів:

$$y = \{y_1, y_2, y_3, ..., y_n\}, \quad (1)$$

Аналітично визначена нелінійна модель, позначено $f(t,c)$, та задана функціональною залежністю або диференціальним рівнянням. З теоретичної точки зору, це найбільш точно описує природу досліджуваного процесу. Необхідно побудувати нелінійну модель досліджуваного процесу $f(t,c)$ за вибіркою експериментальних даних (1).

Далі, використовуючи обраний статистичний алгоритм згладжування, наприклад МНК, формуються параметри експериментальної моделі (2).

$$z\hat{}(t) = \Phi(y), \quad (2)$$

де Ф позначає операції алгоритму визначення параметрів моделі $z\hat{}(t)$ за вибіркою експериментальних даних (1) відповідно до обраного методу статистичного аналізу. Форма моделі $z\hat{}(t)$ є, як правило, поліноміальною в обраному базисі, що має властивості і недоліки традиційних підходів статистичного навчання:

$$y(x) = \sum_{i=1}^{m} c_i \varphi_i(x_i), \quad (3)$$

де c_i – коефіцієнти полінома; $\varphi_i(x_i)$– базисні функції полінома.

Таким чином отримано дві різні моделі. $f(t,c)$ є теоретичною формою, яка більш адекватно відображає властивості досліджуваного процесу, але для неї не відомі ключові параметри $c = \{c_i\}, i = 1 \dots m$. В той час $z\hat{}(t)$ є апроксимуючою експериментальною моделлю, форма і параметри якої відомі з використання класичних статистичних методів, але вона менш адекватна процесу та містить більший вплив випадкових помилок. Операція перенесення властивостей з однієї моделі на іншу є наближенням, реалізувати який можна методом балансу диференціальних спектрів (БДС) [2].

Сутність наближення експериментальної та теоретичної моделей полягає у визначенні параметрів більш складної нелінійної моделі за спрощеною поліноміальною. Для цього можливо використовувати диференційні перетворення з різними критеріями наближення (в альтернативу БДС) [3].

$$\delta(c) = D\left[\left|P\left[\hat{z}(t)\right|_{\hat{t}} \Rightarrow \hat{Z}(k)\right| - \left[P\left[f(t,c)\right|_{\hat{t}} \Rightarrow F(k,c)\right]\right] = D\left[P\left[\varepsilon(t,c)\right|_{\hat{t}} \Rightarrow E(k,c)\right] \rightarrow min, \quad (4)$$

де D позначає операції обраного критерію наближення експериментальної моделі до теоретичної. Визначення параметрів $c = \{c_i\}$ нелінійної моделі $f(t,c)$ здійснюватиметься шляхом формування та розв'язку системи рівнянь:

$$\frac{d\delta(c)}{dc} = 0, \quad (5)$$

Покладемо, що процес, який має експоненційну природу заданий у формі набору вимірів (1) має загальну аналітичну форму:

$$f(t,a) = a_0 + a_1 t + a_2 e^{a_3 t}, \quad (6)$$

де t – це залежний параметр моделі; a – це вектор незалежних параметрів моделі.

Надалі проводиться згладжування вибірки (1) статистичним методом - МНК.

Визначення невідомих параметрів апроксимуючої функції за МНК реалізується за матричним виразом [2]:

$$C = \left(A^T A\right)^{-1} A^T Y, \quad (7)$$

де Y – вектор вибірки виміряних даних; A – матриця базисних функцій; C – вектор шуканих параметрів моделі.

Результатом застосування (7) є набір незалежних параметрів експериментальної моделі, статистично «навчені» за вибіркою вимірів. Надалі визначаються параметри нелінійної моделі за параметрами поліноміальної за схемою (5):

$$V(t,c) = c_0 + c_1 t + c_2 t^2 + c_3 t^3, \quad (8)$$

$$F(t,a) = \left(a_0 + a_2\right) + \left(a_1 + a_2 + a_3\right)t + \frac{a_2 a_3^2}{2}t^2 + \frac{a_2 a_3^3}{6}t^3, \quad (9)$$

де V, F поліноміальні моделі, утворені з диференціальних Р-спектрів початкових моделей.

Наближення двох моделей реалізується за методом БДС, що дає можливість утворити систему рівнянь:

$$\begin{cases} c_0-(a_0+a_2)=0, \\ c_0-(a_1+a_2+a_3)=0, \\ c_0-\frac{a_2 a_3^{\,2}}{2}=0, \\ c_0-\frac{a_2 a_3^{\,3}}{6}=0. \end{cases} \quad (10)$$

Розв'язком системи рівнянь (10) є параметри нелінійної моделі (6), пов'язані із параметрами поліноміальної моделі (8)

$$\begin{cases} a_0=c_0-a_2, \\ a_1=c_1-(a_2*a_3), \\ a_2=\frac{2c_2}{a_3^{\,2}}, \\ a_3=\frac{3c_3}{c_2}. \end{cases} \quad (11)$$

Для перевірки та порівняння якості отриманих результатів з іншими методами, було виконано моделювання з наступними параметрами:

Ідеальний тренд:

$f(x) = 0.0000005x^2, x \in 0,1 \dots 10000;$

Стохастичні відхилення: нормальний закон, $\mu = 0, \sigma = 5;$

Аномальні відхилення: нормальний закон, $\mu = 0, \sigma = 15;$

На рис. 1 візуалізовано ідеальний тренд $f(x)$ з різними відхиленнями позначеними різними кольорами.

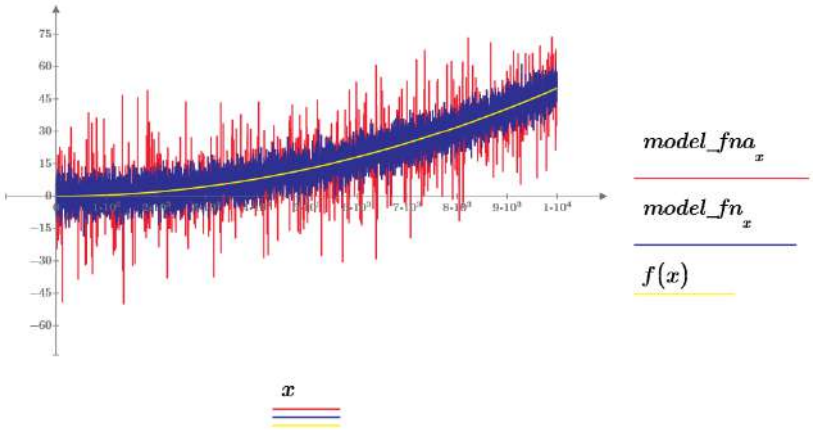


Рис. 1

До обробки моделі вхідних даних – визначення лінії тренду було застосовано три підходи:

- МНК - 1;
- запропонований метод - 2;
- чисельний метод з використанням алгоритму Левенберга-Маркардта -3;

Отримані результати кожного з методів були оцінені за показниками: середньоквадратичне відхилення (СКВ), лінійного відхилення (ЛВ) та часу обрахунку T. Значення показники занесено в табл. 1.

Таблиця 1

Методи	СКВ	ЛВ	T(мкс)
1	37.842	0.102	436
2	44.489	0.098	568
3	37.808	0.102	4647

Висновки. Отримані дані розрахунків та моделювання показують, що застосування розробленого методу для визначення параметрів нелінійної моделі є ефективним за показниками часу та адекватності моделі, порівняно із чисельними підходами. Запропонований підхід базується на використанні методу диференціальних перетворень та

може бути маштабований для інших типів нелінійних моделей. Цей перехід дозволяє покращити також прогностичні характеристики моделі.

Список інформаційних джерел

1. Chapra, Steven C., and Raymond P. Canale. Numerical methods for engineers. Vol. 1221. New York: Mcgraw-hill, 2011. – 987p.
2. Пухов Г.Е. Диференциальные преобразования и математическое моделирование физических процессов. – Киев: Наук. Думка, 1986. – 159 с.
3. Hastie, Trevor, et al. The elements of statistical learning: data mining, inference, and prediction. Vol. 2. New York: springer, 2009. – 764

УДК 004.021:004.9

Мягкий Михайло Юрійович, *аспірант кафедри інформаційних систем та технологій КПІ ім. Ігоря Сікорського, Україна, ORCID ID: https://orcid.org/0000-0002-8038-8839*
Науковий керівник: Гавриленко Олена Валеріївна, *кандидат фізико-математичних наук, доцент, доцент кафедри інформаційних систем та технологій, КПІ ім. Ігоря Сікорського, Україна, ORCID ID: https://orcid.org/0000-0003-0413-6274*

Інформаційна система для аналізу впливу публікацій експертів на курс криптовалютних обмінів на основі багато-агентного підходу

Information system for analyzing the impact of expert publications on the cryptocurrency exchange rate based on a multi-agent approach

Анотація. В даній роботі наведено концепцію інформаційної системи для спрощення процедури моніторингу і прогнозування курсів криптовалюти враховуючи публікації групи експертів в соціальних мережах. Запропоновано багато-агентний підхід для аналізу впливу публікацій експертів у соціальних мережах на курс криптовалютних обмінів. Підхід полягає у моделюванні експертів як агентів з різними рівнями експертизи, наукових досягнень та досвідом у прогнозуванні тенденцій фінансового ринку. Обрані експерти мають задовольняти ряду сформульованих вимог. Робота висвітлює проблему вибору експертів для дослідження впливу їх публікацій на курс криптовалютних обмінів та пропонує розв'язання за допомогою багато-агентного підходу. Дослідження надає інформацію щодо аналізу впливу публікацій експертів на курс криптовалютних обмінів, що може бути корисно для інвесторів та дослідників у цій галузі.
КЛЮЧОВІ СЛОВА: Інформаційна система, криптовалюти, експерти, соціальні мережі, багато-агентний підхід, аналіз впливу публікацій, курс криптовалютних обмінів.

Abstract. This paper presents the concept of an information system to simplify the monitoring and forecasting procedure of cryptocurrency exchange rates, taking into account the publications of a group of experts on social media. A multi-agent approach is proposed to analyze the impact of expert publications on cryptocurrency exchange rates on social media. The approach involves modeling experts as agents with different levels of expertise, scientific achievements, and experience in predicting financial market trends. The selected experts must satisfy a set of formulated requirements. The paper highlights the problem of selecting experts to study the impact of their publications on cryptocurrency exchange rates and proposes a solution using a multi-agent approach. The research provides information on the analysis of the impact of expert publications on cryptocurrency exchange rates, which may be useful for investors and researchers in this field.
KEY WORDS: Information system, cryptocurrency, experts, social media, multi-agent approach, analysis of publication impact, cryptocurrency exchange rate.

Вступ. Останнім часом криптовалюти стали дуже популярними серед інвесторів. Волатильність курсу криптовалют є предметом зацікавлення для багатьох дослідників у галузі фінансів та інвестицій. Один з чинників, які можуть впливати на курс криптовалют, є публікації експертів у соціальних мережах, що може бути підтверджено за домогою кореляційного аналізу [1,2]. Завдання дослідження впливу публікацій експертів на курс криптовалютних обмінів вимагає постійного моніторингу як курсів криптовалют, так і дописів у соціальних мережах. Для спрощення цієї задачі розробляється інформаційна система, яка буде вирішувати проблему моніторингу змін курсів криптовалюти і нових публікацій у соціальних мережах, а також побудови якісних прогнозів на основі цих даних. Дана

система базується на багато-агентному підході і здатна зібрати, обробити та проаналізувати дані з різних джерел. Таким чином, основною задачею яка виникає при взаємодії з інформаційною системою є вибір експертів для дослідження впливу їх публікацій на курс криптовалют, що є складною задачею. Це є необхідним для подальшого отримання найбільш точних прогнозованих значень курсів криптовалют.

Попередньо було проведено дослідження на основі одноагентної системи з вибором єдиного експерта, концепція даного методу було наведено в статті [3]. Ефективність наведеного алгоритму, а саме вплив дописів на курс криптовалюти може бути оцінений за допомогою кореляційного аналізу.

Подальшим етапом дослідженням є визначення “основного” експерта за допомогою обчислення найбільшої апостеріорної ймовірності, того, що якщо в будь-який момент часу з прогнозованого інтервалу, буде опублікований пост пов’язаний з досліджуваною криптовалютою, то його автором буде “основний” експерт.

Отримані апостеріорні ймовірності можуть надалі використовуватися для знаходження середніх оцінок ефективності прогнозуючих адаптивних алгоритмів зміни курсу криптовалют під впливом послідовної появи у часі індивідуальних чи групових постів експертів. Створення таких алгоритмів та, як наслідок, відповідних інтелектуальних технологій є предметом подальших досліджень авторів.

Для вирішення поставленої задачі запропоновано використовувати підхід з багатьох агентів для аналізу впливу публікацій експертів у соціальних мережах на курс криптовалютних обмінів. Запропоновано моделювати експертів як агентів з різними рівнями експертизи, наукових досягнень та досвідом у прогнозуванні тенденцій фінансового ринку.

Визначення поняття інформаційна система.

Інформаційна система - це сукупність комп'ютерних програм, що обробляють інформацію для підтримки різних процесів та діяльності. Інформаційна система складається з апаратного та програмного забезпечення, баз даних та спеціалізованих програм для різних завдань. Вона забезпечує збір, обробку, збереження та передачу інформації в режимі реального часу або в заданий період часу [4].

Інформаційні системи можуть бути різними за своєю призначеністю та масштабом, від невеликих систем для індивідуального користувача до великих корпоративних систем, що охоплюють великі об'єкти, такі як компанії, урядові організації та національні економіки.

Інформаційні системи використовуються в багатьох галузях діяльності, таких як управління бізнесом, наука, освіта, медицина, транспорт та багато інших. Вони є важливим інструментом для підтримки різних видів діяльності, таких як планування, управління, моніторинг, аналіз, формування рекомендацій та прийняття на їх основі рішень стосовно проблематики поставленої задачі.

Відповідність розробленої системи вимогам до інформаційних систем.

В системі є два головних актори:

- Розробник - займається впровадженням і підтримкою системи;
- Користувач - використовує систему для отримання рекомендацій стосовно курсів криптовалют.

Сценарій роботи розробленої рекомендаційної системи складається з наступних етапів:

- Підготовчий етап - внесення даних про обрані криптовалюту та соціальну мережу, а також формування критеріїв для вибору групи експертів і відбору їх публікацій. Після чого визначається часовий проміжок на котрому збираються дані про курси обраної криптовалюти;
- Етап аналізу - згідно визначених критеріїв формується експертна група і збираються дані про їх публікації, які будуть відповідати сформованим вимогам. Після чого проводиться їх сентиментальний аналіз і досліджується рівень впливу на курс криптовалют. Результатом даного етапу буде формування прогнозу на основі отриманих даних;
- Етап контролю якості прогнозу - проводиться розрахунок міри точності

прогнозу та інтерпретація отриманих результатів.

Даний сценарій рекомендаційної системи наведено на рисунку 1.

Згідно цього сценарію бачимо, що система дійсно відповідає висунутим вимогам до інформаційних систем, тому і є інформаційною системою.

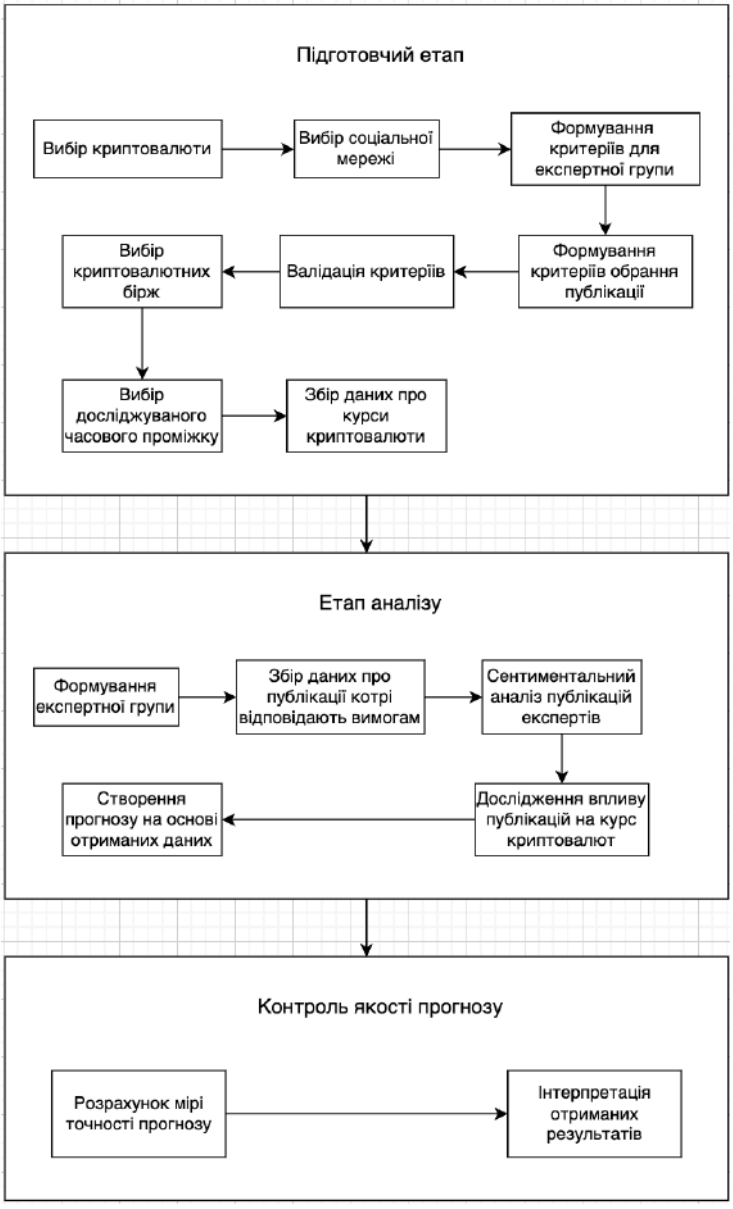


Рис. 1. Сценарій роботи інформаційної системи для прогнозування курсів криптовалют з урахуванням впливу публікацій групи експертів

Формування вимог до експертів і їх публікацій в соціальних мережах.

Для вирішення поставленої задачі з прогнозування курсів криптовалютних обмінів з урахуванням впливу публікацій експертів необхідно сформувати перелік критеріїв яким вони мають відповідати.

- Експерт має бути публічною особою;
- кожен з експертів має бути активним користувачем обраної соціальної мережі, а також мають велику аудиторію підписників;
- всі експерти належать до різних професійних кіл;
- основна професійна діяльність експертів так чи інакше пов’язана з використанням криптовалюти;
- кожна пара експертів, не підтримує спілкування в обраній соціальній

мережі (не є друзями та не реагують на пости одне одного);
- мати достатній кваліфікаційний рівень в фінансовій сфері;
- мати досвід експертизи фінансового ринку.

Даний перелік дозволить формувати групи експертів таким чином, щоб до них не входили публічні особистості, котрі хочуть завдяки популярності вплинути тим чи іншим чином на курс криптовалюти за досліджуваний період з метою не пов'язаною з їх основним видом діяльності.

До публікацій експертів також має бути сформований ряд критеріїв, таких як:

- серед публікацій конкретного експерта виділяються публікації в яких є пряма чи непряма згадка зазначеної криптовалюти;
- публікації мають належати досліджуваному відрізку часу;
- публікації обираються з зазначеної соціальної мережі;
- публікації мають містити текстовий зміст.

Дані критерії дозволяють коректно провести розрахунок коефіцієнта рівня впливу публікації на курс криптовалюти.

Висновки. Отримані результати свідчать про наявність істотного впливу публікацій експертів при прогнозуванні курсу криптовалютних обмінів [1].

Вибір експертів є складної задачею при дослідженні впливу їх публікацій у соціальних мережах на курс криптовалютних обмінів. З огляду на це використання багато-агентного підходу для розв’язання даної задачі є ефективним рішенням, що було підтверджено статистичним шляхом. Багато-агентний підхід надає найбільш наближені до реального курсу прогнозовані значення через моделювання взаємодії між агентами та аналізуючи вплив публікацій експертів на курс криптовалютних обмінів.

Задача дослідження впливу публікацій експертів на курс криптовалютних обмінів вимагає постійного моніторингу як курсів криптовалют так і дописів в соціальних мережах. Для спрощення цієї задачі і розробляється дана інформаційна система.

Отже, використаний алгоритм в даній інформаційній системі використовується для розробки інструменту, який може допомогти інвесторам та дослідникам в аналізі впливу публікацій експертів на курс криптовалютних обмінів.

Список інформаційних джерел

1. Гавриленко О.В., Мягкий М.Ю. Дослідження впливу публікацій відомих людей на курс криптовалют. Збірник тез VI Міжнародної науково-практичної конференції “Прикладні системи та технології в інформаційному суспільстві”, 2022. С. 51-55.

2. Мягкий М.Ю., Гавриленко О.В. Огляд задачі аналізу публікацій та розроблення методів їх розв’язання. Перша Всеукраїнська науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021) : Матеріали всеукраїнської науково-практичної конференції молодих вчених та студентів, м. Київ, 22 листоп. 2021 р. – 26 трав. 2022 р. Київ, 2021. С. 195-198.

3. O. Gavrylenko, M. Miahkyi, Y. Zhurakovskyi. The task of analyzing publications to build a forecast for changes in cryptocurrency rates // Adaptive automatic control systems. Vol. 2 No. 41. - K.: NTUU "KPI", 2022. pp. 90-99. DOI: https://doi.org/10.20535/1560-8956.41.2022.271349.

4. Sebastian K Boell, Dubravka Cecez-Kecmanovic (2015), “What is an Information System?”, Conference: Proceedings of the 48th Hawaiian International Conference on System Sciences (HICSS). DOI:10.1109/HICSS.2015.587.

УДК 004.414

Калашник Роман Сергійович, *здобувач вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник*: *Крилов Євген Володимирович,
кандидат наук, доцент, доцент кафедри інформаційних систем та технологій КПІ ім. Ігоря Сікорського, Україна

СИСТЕМА ІНІЦІАЛІЗАЦІЇ ТА УПРАВЛІННЯ КОНТЕЙНЕРАМИ І ЗОВНІШНІМИ СЕРВІСАМИ У KUBERNETES КЛАСТЕРІ

Анотація. Ці тези описують принципи ініціалізації та управління контейнерами і зовнішніми сервісами у Kubernetes кластері. У тезах описано ключові компоненти системи, розглядаються основні методи управління ресурсами та наведений короткий опис розробленої системи.
КЛЮЧОВІ СЛОВА: Kubernetes, контейнери, сервіси, ініціалізація, управління, Gitlab, Docker, AWS, Terraform.

Abstract. These theses describe the principles of initializing and managing containers and external services in a Kubernetes cluster. The thesis describes the key components of the system, discusses the main methods of resource management, and provides a brief description of the developed system.ц
KEY WORDS: Kubernetes, containers, services, initialization, management, GitLab, Docker, AWS, Terraform.

Вступ. З розвитком мікросервісної архітектури та контейнерізації, системи оркестрації контейнерів, такі як Kubernetes, стали важливим інструментом для розгортання та управління застосунками. Kubernetes дозволяє автоматизувати процеси ініціалізації, моніторингу та масштабування контейнерів, а також спрощує інтеграцію зовнішніх сервісів. Ці тези досліджують принципи ініціалізації та управління контейнерами і зовнішніми сервісами у Kubernetes кластері а також надається короткий опис розробленої системи.

Основна частина.

1. Архітектура Kubernetes.

Kubernetes базується на концепції кластерів, які складаються з вузлів (nodes) - фізичних або віртуальних машин. Вузли виконують контейнери (containers), які є запакованими застосунками з усіма необхідними залежностями. Контейнери згруповані в поди (pods), що є найменшими розгорнутими одиницями у Kubernetes.

Контролери реплікації (replica sets) стежать за кількістю реплік подів, що запущені в кластері, і відповідають за масштабування та відновлення. Сервіси (services) надають абстракцію для доступу до подів, дозволяючи взаємодію між контейнерами та зовнішніми сервісами.

Кластери Kubernetes контролюються майстром (master), який включає API-сервер, контроллер-менеджер, планувальник (scheduler) та etcd - розподілене сховище конфігураційних даних.

2. Ініціалізація контейнерів і зовнішніх сервісів.

Ініціалізація контейнерів та зовнішніх сервісів в Kubernetes кластері полягає в налаштуванні ресурсів перед запуском основних контейнерів. Ініціалізаційні контейнери (Init Containers) використовуються для виконання задач, які повинні бути завершені перед запуском основних контейнерів. Це може включати налаштування мережі, завантаження даних або перевірку доступності зовнішніх сервісів.

Конфігураційні файли та секрети (secrets) дозволяють забезпечити контейнери конфігураційними даними та чутливою інформацією, наприклад, паролі або ключі доступу. Зовнішні сховища даних, такі як Persistent Volumes (PV) та Persistent Volume

Claims (PVC), використовуються для зберігання даних, які повинні вижити після видалення подів.

3. Управління контейнерами і зовнішніми сервісами.

Управління контейнерами та зовнішніми сервісами у Kubernetes кластері здійснюється за допомогою керуючих об'єктів, таких як Deployment, StatefulSet, DaemonSet та CronJob.

Deployment відповідає за розгортання та оновлення подів, а також за горизонтальне масштабування.

Stateful Set використовується для розгортання застосунків, які потребують постійного зберігання або забезпечення стабільності мережевих ідентифікаторів.

DaemonSet гарантує, що на кожному вузлі кластера запущено одну копію заданого пода.

CronJob дозволяє запускати періодичні задачі в контейнерах за визначеним розкладом.

4. Сервіси та мережеві ресурси.

Сервіси та мережеві ресурси у Kubernetes спрощують взаємодію між контейнерами та зовнішніми сервісами.

Сервіси використовують різні типи експозиції: ClusterIP створює внутрішній IP-адрес у кластері, доступний лише з кластера. NodePort відкриває порт на кожному вузлі кластера, що дозволяє доступ ззовні.

Load Balancer автоматично створює зовнішній балансувальник навантаження для сервісу, який перенаправляє трафік на внутрішній ClusterIP або NodePort.

Ingress дозволяє управляти доступом до сервісів на рівні HTTP/HTTPS, використовуючи правила маршрутизації та інші додаткові функції, такі як TLS-шифрування та аутентифікація.

5. Gitlab.

GitLab - це веб-платформа для керування проектами з відкритим вихідним кодом, яка базується на системі контролю версій Git. GitLab надає середовище для співпраці розробників, яке включає репозиторії вихідного коду, систему стеження за помилками, інструменти для рецензування коду, систему безперервної інтеграції та безперервної розгортки (CI/CD), вікі для документації та інші можливості для полегшення роботи розробників.

GitLab доступний у двох версіях: безкоштовній (GitLab Community Edition, CE) та комерційній (GitLab Enterprise Edition, EE). GitLab Community Edition надає основні функції для керування кодом та співпраці, тоді як GitLab Enterprise Edition містить додаткові можливості для великих організацій та корпоративних команд, такі як підтримка більшої кількості користувачів, преміальна підтримка та покращена безпека.

GitLab дозволяє розробникам ефективно створювати, змінювати та відстежувати зміни у коді, від початку розробки до деплою, а також працювати над проектами з командами розробників різних розмірів і з різними рівнями досвіду.

6. Docker.

Docker - це платформа для розробки, розгортання та тестування застосунків у контейнерах. Контейнери дозволяють розробникам упаковувати застосунок з усіма його залежностями (бібліотеки, фреймворки, конфігураційні файли тощо) в єдиний стандартний пакет, який можна легко розгорнути на будь-якій системі, що підтримує Docker.

Docker використовує віртуалізацію на рівні операційної системи, яка дозволяє декільком контейнерам працювати на одному фізичному хості, ізолюючи ресурси та середовища один від одного. Це робить контейнери легшими та швидшими, ніж традиційні віртуальні машини, оскільки вони не вимагають окремої операційної системи для кожного контейнера.

7. Terraform.

Terraform - це інструмент для інфраструктури як код (Infrastructure as Code, IaC), розроблений компанією HashiCorp. Він дозволяє користувачам визначати, створювати та управляти інфраструктурою обчислювальних середовищ за допомогою простих текстових конфігураційних файлів. Terraform може працювати з різними хмарними провайдерами, такими як AWS, Azure, Google Cloud Platform, а також з іншими сервісами та платформами.

8. Система ініціалізації та управління контейнерами і зовнішніми сервісами у Kubernetes кластері. Для процессу розробки було використано: Gitlab, Docker, Terraform, AWS, Gitlab Runners, Gitlab CI/CD, Gitlab Registry, Global Secret Values, Docker self-contained images.

Система побудована на основі платформи Gitlab і поділяється на 3 підсистеми а саме:

підсистема ініціалізації та управління зовнішніми сервісами та Kubernetes кластером, підсистема ініціалізації контейнерів та підсистема управління контейнерами.

Підсистема ініціалізації та управління зовнішніми сервісами та Kubernetes кластером працює на основі поєднань таких сервісів як: Gitlab, Gitlab CI/CD, Gitlab Runner, Docker, Terraform, Secret values, AWS. Вона ініціалізує, модифікує та керує усіма змінами та модифікаціями що, стосуються кластеру та сервісами. Мови програмування YAML, HCL.

Підсистема ініціалізації контейнерів працює на основі поєднань таких сервісів як: Gitlab, Gitlab CI/CD, Gitlab Runner, Gitlab Registry, Docker, Secret Values та AWS CLI. Вона ініціалізує та оновлює контейнери та внутрішні сервіси кластера для функціонування контейнерів. Мови програмування YAML, BASH.

Підсистема управління контейнерами працює на основі поєднань таких сервісів як: Gitlab, Gitlab CI/CD, Gitlab Runner, Docker, AWS CLI, Secret Values та kubectl. Вона виконує функцію управління контейнерам та внутрішніми сервісами кластеру. Мови програмування YAML, BASH.

Висновки. Система ініціалізації та управління контейнерами і зовнішніми сервісами у Kubernetes кластері дозволяє автоматизувати та спрощувати процеси розгортання, моніторингу та масштабування застосунків. Основні компоненти, такі як поди, сервіси та керуючі об’єкти, сприяють гнучкості та стабільності системи. Ця стаття досліджує різні методи ініціалізації, управління ресурсами та використання мережевих ресурсів для підтримки взаємодії між контейнерами та зовнішніми сервісам у кластері.

Список інформаційних джерел

1. Kubernetes Documentation: https://kubernetes.io/docs/
2. Burns, B., Beda, J., & Hightower, K. // Kubernetes: Up and Running: Dive into the Future of Infrastructure. -2017. -B.O’Reilly Media.
3. Poulton, N. // Kubernetes for Developers: Use Kubernetes to Develop, Test, and Deploy Your Applications with Help from Containers. -2019. -B.Packt Publishing.
4. Chalkley, N. // Kubernetes Cookbook: Practical Solutions to Container Orchestration. -2018. -B.Packt Publishing.
5. Swoods, S., & Murugesan, S. // Mastering Kubernetes: Level up your container orchestration skills with Kubernetes to build, run, secure, and observe large-scale distributed apps. -2019. -B.Packt Publishing.
6. Bilgin I., Roland H. // Red Hat, Kubernetes Patterns: Reusable Elements for Designing Cloud-Native Applications. -2019. -B.O`Reilly.
7. GitLab Documentation: https://docs.gitlab.com/
8. Docker Documentation: https://docs.docker.com/
9. Terraform Documentation: https://developer.hashicorp.com/terraform/doc

УДК 004.415.2; 004.422

Михалік Єлизавета Ігорівна, *здобувач вищої освіти*
ХНУРЕ, Україна
Українська Оксана Олександрівна, *здобувач вищої освіти*
ХНУРЕ, Україна
Науковий керівник: Широкопетлева Марія Сергіївна, *старший викладач кафедри Програмної інженерії, ХНУРЕ, Україна*

ВИКОРИСТАННЯ GOLANG ДЛЯ РОЗРОБКИ BACKEND СКЛАДОВОЇ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ МЕРЕЖІ ЗАРЯДНИХ СТАНЦІЇ ЕЛЕКТРОМОБІЛІВ

Анотація. В роботі представлено основний функціонал для програмної системи підтримки роботи мережі зарядних станцій для електромобілів та показано доцільність використання сучасної мови програмування GoLang для розробки Backend складової програмної системи для системи зарядних станції електромобілів та переваги цієї мови.
КЛЮЧОВІ СЛОВА: GOLANG, BACKEND, ПРОГРАМНА СИСТЕМА.

Abstract. The paper presents the main functionality for a software system to support the operation of a network of charging stations for electric vehicles and shows the feasibility of using the modern programming language GoLang to develop the Backend component of a software system for an electric vehicle charging station system and the advantages of this language.
KEY WORDS: GOLANG, BACKEND, SOFTWARE SYSTEM.

Вступ. У сучасних реаліях все більшої популярності набувають електромобілі. Кожна велика автомобільна компанія має у своїй лінійці електромобіль, адже це вони мають багато плюсів у порівнянні з дизельними автомобілями і головна перевага – це мінімізація забруднення навколишнього довкілля. У якості топлива ці автомобілі використовують електроенергію, яку отримують на спеціальних зарядних станціях. Мережа цих станцій швидко шириться у нашій країні та світі, саме тому виникає задача у програмних системах, що дозволяють обслуговувати ці станції та взаємодіяти їм з користувачем. Розробка та підтримка роботи цих станцій створює нові виклики, а саме: стандартизація, управління навантаженням, обробка великих об’ємів даних та зручність використання для середньостатистичного користувача.

Основна частина. Мова програмування Go, розроблена компанією Google, стає все більш популярною в останні роки. Однією з основних переваг цієї мови є простота та зрозумілість її синтаксису, що дозволяє легко писати, тестувати та підтримувати величезні об’єми коду, її ефективність, зокрема, її швидкодія і ефективне використання ресурсів. Ці переваги особливо важливі для програмної систем для системи зарядних станцій для електромобілів, які потребують швидкої і надійної обробки великих об’ємів даних.

Backend частина програмної системи для системи зарядних станцій для електромобілів - це ключовий елемент системи, який забезпечує зв'язок між Frontend складовою (інтерфейсом користувача) та базою даних, яка містить інформацію про зарядні станції, їх розташування, електромобілі та користувачів, а також дозволяє легко інтегрувати платіжну систему для оплати електроенергії. Для розробки такої системи потрібна мова програмування, яка забезпечує швидку та ефективну обробку запитів від фронтенду та доступ до бази даних. Golang - це ідеальний вибір для цієї задачі.

У ході розробки такої програмної системи виникає декілька ключових задач:

- отримання запитів від Frontend складової;
- безпечна, надійна та швидка обробка і передача даних;
- можливість інтегрувати платижну систему;
- можливість моніторинга місцезнаходження зарядної станції та самого процесу зарядки зі сторони користувача автомобіля;
- можливість зберігання інформації щодо різних зарядних станцій

Для ефективного рішення цих ключових задач пропонується використовувати мікросервісну архітектуру [1], що поділяє усю систему на сукупність невеличких сервісів, кожний з яких працює у власному процесі та спілкується з рештою сервісів, використовуючи HTTP протокол передачі даних. Ця архітектура надає можливість грамотно розподіляти ресурси між сервісами, ти самим забезпечуючи швидку роботу всієї системи, а також швидко масштабувати систему, без значних змін, що є дуже важливою перевагою у контексті розробки програмної системи для системи зарядних станції – інфраструктури, що росте у геометричній прогресії.

Далі визначимо один з варіантів стеку технології, які можуть використовуватися для імплементації цієї системи:

- мова GoLang;
- реляційна СУБД MySQL;
- IoT пристрій на основі ESP32;
- платіжна система Liqpay

Мова GoLang має усі необхідні вбудовані засоби для взаємодії з цим стеком технології, а також для інтеграції IoT пристроя ESP32 [2]. Також ця мова має ефективні та прості засоби тестування розробленої системи, що дозволяє впевнитись в тому, що система надійна та працює коректно.

Наведемо загальну діаграму активностей (рис.1) програмної системи для системи зарядних станцій для електромобілів, що відображає ключові моменти програмної системи.

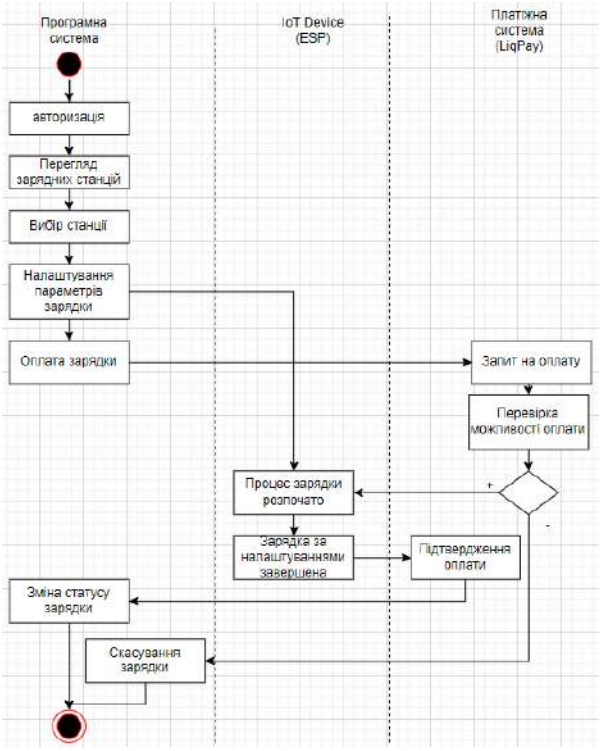


Рисунок 1 – Загальна діаграма активностей

Frontend взаємодіє з backend частиною через HTTP запити та дає можливість виконувати наступні дії, що перераховані далі. На діаграмі показано, що Backend частина, реалізована на Golang, дозволяє користувачу авторизуватись в системі, переглянути список доступних зарядних станцій, вибрати потрібну, задати налаштування зарядки (потужність, кількість ват) та перейти безпосередньо до етапу оплати. Гілки для обробки різних сценаріїв, таких як помилка аутентифікації, і авторизація платежу узагальнені. Налаштування сеансу зарядки мають бути передані як в платіжну систему так і на IoT-пристрій для подальшої роботи. Для оплати послуг зарядних станції використовується платіжна система Liqpay, яку мова GoLang дозволяє легко інтегрувати. Після перевірки того, що користувач на своїй картці має достатню кількість коштів, передається сигнал на IoT пристрій ESP32 та розпочинається процес зарядки. Після успішного завершення процесу користувач, отримує повідомлення про це.

У разі виникнення помилок під час виконання запиту, система повертає відповідний код помилки, який дозволяє Frontend частині відобразити відповідне повідомлення користувачу про причину невдачі. Для забезпечення безпеки та захисту персональних даних користувачів, використовується протокол HTTPS для передачі даних між клієнтом та сервером. Безпека також забезпечується за допомогою автентифікації та авторизації користувачів.

Усі ці дії виконуються з використанням мови програмування Golang, яка дозволяє швидко та ефективно обробляти великий потік даних та запитів, що є важливим для ефективної роботи системи зарядних станцій. Таким чином, використання мови програмування Golang для розробки Backend частини програмної системи для системи зарядних станцій для електромобілів дозволяє покращити ефективність та швидкість роботи системи, а також забезпечити більш точний та швидкий моніторинг стану зарядних станцій.

Крім того, мова програмування Golang має вбудовану підтримку конкурентного програмування, що дозволяє розробляти багатопоточні та розподілені застосунки. Це особливо важливо для системи зарядних станцій, де може бути багато одночасних запитів на зарядку електромобілів.

Висновки. В роботі розглянуто використання сучасної мови програмування GoLang та її переваги у контексті реалізації Backend частини програмної системи для системи зарядних станцій для електромобілів. Як можна побачити з практичного прикладу, GoLang забезпечує не лише простоту та ефективність для розробників систем, а й безпечність, швидкість передачі даних для користувача. Галузь зарядних електростанцій для електромобілів потребує сучасних рішень, що сприятимуть її розвитку та масштабуванню і одним з найефективніших інструментів у створенні таких рішень є мова програмування GoLang.

Список інформаційних джерел

1. What are the benefits of a microservices architecture? URL: https://about.gitlab.com/blog/2022/09/29/what-are-the-benefits-of-a-microservices-architecture/ / GitLab // GitLab DevOps Platform Blog. – 2022.

2. Using Go language to program embedded devices.URL: https://medium.com/vacatronics/lets-go-embedded-with-esp32-cb6bb3043bd0 / Fernando Souza // Medium.org. – 2020.

УДК 004.7

Пономаренко Павло Анатолійович, *здобувач вищої освіти*
Дніпровського національного університету імені Олеся Гончара, Україна
Науковий керівник: Сидорова Марина Геннадіївна, *кандидат технічних наук, доцент кафедри математичного забезпечення ЕОМ Дніпровського національного університету імені Олеся Гончара, Україна*

СТВОРЕННЯ КЛІЄНТА БЛОКЧЕЙН МЕРЕЖІ МОВОЮ JAVA

Анотація. Проєктування та розробка клієнта децентралізованої облікової мережі включають широкий спектр задач, пов'язаних із реалізацією криптографічних операцій, хешування даних, передачею та зберіганням транзакцій, майнінгом нових блоків. Приклад проєкту Cryptocoin демонструє підхід до створення клієнта блокчейн мережі засобами Java, Spring Boot, PostgreSQL.

КЛЮЧОВІ СЛОВА: блокчейн, криптогафія, клієнт мережі, майнінг

Abstract. Design and development of a client of a decentralized accounting network include a wide range of tasks related to cryptographic operations implementation, data hashing, transactions transferring and storing, new blocks mining. The example of Cryptocoin project demonstrates the approach for creating a blockchain network client using Java, Spring Boot, and PostgreSQL.

KEY WORDS: blockchain, cryptography, network client, mining.

Вступ. Головною метою створення клієнта Cryptocoin була реалізація базового функціоналу, який надає популярним блокчейн мережам (Bitcoin, Ethereum, Cardano та інш.) наступні властивості: незалежність від централізованого контролю, trustless (відсутність будь-якої довіри між вузлами мережі), permissionless (вільне використання), fault tolerance (відмовостійкість). Розробка велася на основі Java Development Kit 16 та Spring Boot.

Основна частина. Функціонал додатку був розбитий за задачами на шість розділів: криптографічні операції, зберігання транзакцій та блоків у реляційних таблицях, прийом і розсилання інформації мережею, управління конфігурацією вузла, валідація транзакцій, майнінг.

У додатку реалізований електронний підпис зі специфікацією ECDSA, в основі якого лежить асиметрична криптографія з відкритим та закритим ключем. Генерація ключів, створення та перевірка підпису виконуються за допомогою арифметичних операцій над скінченним полем точок еліптичної кривої: додавання, подвоєння, множення на скаляр, обернення. У проєкті використовується еліптична крива secp256r1. Закритий ключ генерується випадково, з нього через множення на базову точку знаходиться відкритий. Значення ключів обробляються і зберігаються в програмі у форматі типу BigInteger, числа пересилаються між вузлами у шістнадцятковому вигляді. Відтворено алгоритм формування та перевірки підпису [1].

Для зберігання інформації у вузлі мережі використовується СУБД PostgreSQL. У базі даних створюються наступні таблиці:

1. blocks — зберігає хеш попереднього блоку, корінь дерева Меркла, час створення, розв'язок ресурсоємної задачі.
2. transactions — зберігає хеш блоку до якого відноситься. Між transactions та blocks є відношення багато-до-одного.
3. vin — зберігає дані про попередній вихід, із якого беруться монети для нової транзакції: хеш транзакції, номер виходу, підпис, ключ до нього. Між vin та transactions є відношення багато-до-одного.
4. vout — зберігає дані про вихід поточної транзакції: сума переказу, адреса отримувача.

Між vout та transactions є відношення багато-до-одного.

5. accounts — зберігає дані про поточні рахунки користувачів: відкритий ключ, сума.

Початковий вміст бази даних (таблиці та стартовий блок) при розгортанні додатку автоматично створюється засобами бібліотеки Liquibase.

Приймання вхідних даних мережі здійснюється через open API інтерфейс, який реалізується класами-контролерами, що приймають HTTP запити від інших вузлів. Клас RestTemplate із бібліотеки Spring використовується для поширення блоків і транзакцій системою. У додатку реалізована тришарова архітектура. Функції та класи розподілені для прийому обробки та зберігання даних.

Блокчейн мережа є графом нерівномірно пов'язаних між собою вузлів. Щоб клієнт знав, через які її учасники він може розповсюджувати інформацію, при запуску jar файлу в параметрах командної строки вказується аргумент *addresses*, в якому перелічуються адреси сусідніх вузлів.

Транзакції, які надсилаються додатку, зберігаються в черзі *mempool*. Вони сортуються та обробляються в паралельному циклі. Для роботи з *mempool* використовуються синхронізовані методи, адже елементи додаються та вилучаються паралельно. Кожна транзакція перевіряється на коректність підписів та відповідність до записів у базі даних. Пріоритет для додавання в блок мають ті елементи черги, які пропонують більші комісійні внески.

У проєкті Cryptocoin реалізовано алгоритм консенсусу proof-of-work. Для того щоб запропонувати новий блок, клієнт повинен знайти хеш фіксованої складності. Процес майнінгу складається з ітеративного перебору варіантів хешу при додаванні до вмісту блока зростаючої змінної *i*, яка в разі успіху виступає доказом розв'язку ресурсоємної задачі. Його можливо досягнути лише методом грубої сили, що захищає систему від спаму і шахрайства. Шуканий хеш є числом і має бути меншим за фіксоване порогове значення. Процес майнінгу переривається, якщо інший вузол надсилає клієнту коректний блок із своїм розв'язком. Якщо клієнт першим досягнув шуканого результату, він нагороджується комісійними монетами [2].

Висновки. Cryptocoin вузол не потребує дозволів на взаємодію з іншими клієнтами мережі, але надійно перевіряє всю вхідну інформацію від них. Описаний функціонал забезпечує децентралізоване саморегулювання системи, що досягається через консенсус її учасників.

Список інформаційних джерел

1. Hankerson D., Menezes A., Vanstone S. Guide to Elliptic Curve Cryptography. NY: Springer New York. 2010. 312p.

2. Bashir Imran Mastering Blockchain Second Edition. Birmingham: Packt Publishing Ltd. 2018. 911p.

УДК 004.414

Барченко Павло Васильович, *здобувач вищої освіти*
Харківський національний університет радіоелектроніки, Україна
Науковий керівник: Мазурова Оксана Олексіївна, *кандидат технічних наук,*
доцент, доцент кафедри програмної інженерії,
Харківський національний університет радіоелектроніки, Україна

ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ІНТЕГРАЦІЇ СЕРВІСІВ

Анотація. Дана робота присвячена оптимізації інтеграції сервісів фреймворку для e-commerce рішень. Розроблена система з 4х сервісів для експериментального тестування показників викликів API. Розглядалося виконання запитів для вибору даних користувачів у декількох варіаціях з використанням механізму кешування та оптимізації механізму багатопоточності у роботі одного з сервісі.
КЛЮЧОВІ СЛОВА: сервісіс, виклик, інтеграція.

Abstract. The given work is devoted to optimizing the integration of framework services for e-commerce solutions. A system of 4 services has been developed for experimental testing of API call indicators. The implemented system has the functions of basic calling of a regular request for selecting user data in several variations using the caching mechanism and optimization of the multithreading mechanism in the work of one with the service.
KEY WORDS: service, call, integration.

Вступ. API (інтерфейси прикладного програмування) стали стрижнем безперебійного зв'язку між різними програмними додатками. Як канали, що полегшують обмін даними, API незамінні для створення надійних цифрових екосистем. В умовах постійно зростаючої залежності від API вкрай важливо оптимізувати їх роботу і уникнути повільного відгуку, що може суттєво утруднити роботу користувачів.

Хороша продуктивність API гарантує, що програми які їх використовують, зможуть забезпечити безперебійну роботу користувачів. Низька продуктивність може призвести до повільного часу відгуку, зниження функціональності або навіть збоїв у роботі програми, що призведе до невдоволення користувачів та втрати клієнтів.. Вирішення проблем затримки та підвищення пропускної спроможності вимагає реалізації ефективних алгоритмів та структур даних, а також оптимізації мережевих протоколів та інфраструктури. Це вимагає від розробників глибокого розуміння базових систем та властивих їм обмежень [1].

Надійність і доступність API є важливими вимогами, оскільки API повинні бути стійкими до збоїв і легко справлятися з різними сценаріями помилок. Це потребує впровадження механізмів відмовостійкості та самовідновлення, а також комплексних систем моніторингу та оповіщення.

Нарешті, задоволення потреб користувачів і забезпечення безперебійної масштабованості вимагає впровадження еластичних і модульних архітектур. Отже, розробники повинні вміти використовувати передові технології та методології, такі як контейнеризація та мікросервіси, для підтримки бездоганної продуктивності API в умовах технологій, що постійно розвивається.

Підвищення продуктивності API вимагає багатогранного підходу, який охоплює різні аспекти життєвого циклу розробки

програмного забезпечення, від проектування та реалізації до моніторингу та оптимізації [1].

Основна частина. Проблема продуктивності API полягає у досягненні тонкого балансу між швидким та надійним наданням послуг та дотриманням обмежень інфраструктури, бюджету та масштабованості. Оскільки API стають все більш невід'ємною частиною сучасних програмних систем, обслуговуючи безліч користувачів та сценаріїв використання, підтримання оптимальної продуктивності стає все більш необхідним [2].

Мета даного дослідження - висвітлити основні методи та кращі практики для підвищення продуктивності API, які тим самим забезпечують швидкий час відгуку для користувача.

Ретельно вивчивши базову архітектуру, розумно використовуючи механізми кешування, оптимізуючі навантаження запитів та відповідей, розробники зможуть по-справжньому використати потенціал API для створення гнучких та ефективних цифрових платформ.

Таким чином, була поставлені наступні задача:

– провести аналіз існуючих методів оптимізації інтеграції сервісів;

– спланувати експериментальну частину дослідження, що включає розробку сервісів та інтеграцій між ними;

– провести експериментальне дослідження методів обробки запитів у мікросервісній архітектурі.

В роботі було розглянуто та проаналізовано наступні методи оптимізації інтеграцій між сервісами при розробці фреймворку для e-commerce рішень:

– використання ефективних алгоритмів та структур даних для зниження складності обчислень, а також багатопоточної обробки запитів;

– балансування навантаження: розподіл вхідних API-запитів між кількома серверами дозволяє усунути вузькі місця та забезпечити стабільну продуктивність у періоди підвищеного попиту;

– кешування: впровадження стратегій кешування, таких як прикордонне кешування або кешування на рівні програми, може значно скоротити час відгуку за рахунок мінімізації надлишкового пошуку та обробки даних [2].

В ході експериментального дослідження було розроблено декілька сервісів, які були застосовані у розробці фреймворку для e-commerce рішень (див. рис. 1). Розглянемо прийняті рішення:

– інтеграція проходить через ESB сервіс, він і є декоратором для зовнішнього світу; тому запити робляться через нього до TDS сервісу;

– деякі запити для замірів робляться через сервіс кешування або на пряму через Bisness logic layer сервіс;

методи оптимізації сервісів досліджувалися шляхом порівняння продуктивності запитів до БД через API; база даних розгорнута у Postgresql:

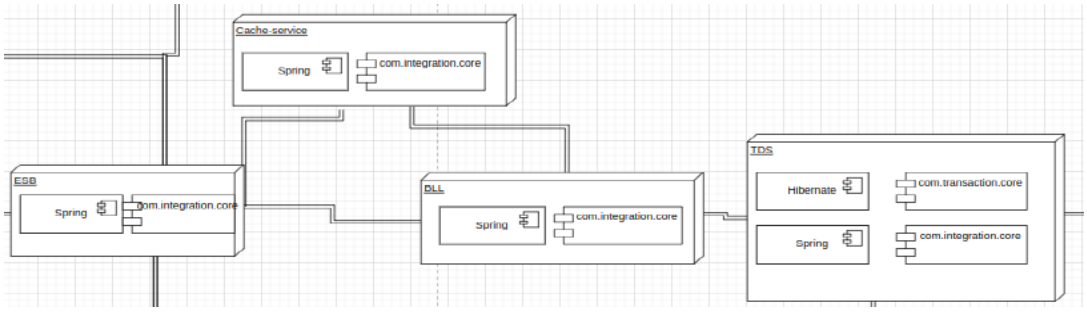


Рисунок 1 – Фрагмент архітектури розробленого фреймворку

На основі розробленого фреймворку було проведено серію експериментів (див. рис. 2), таких як:
– перший виклик базового API сервісу тривав 219 ms;
– багаторазовий виклик базового API тривав сервісу 21 ms;
– перший виклик сервісу з багатопоточною оптимізацією тривав 21ms;
– багаторазовий виклик сервісу з багатопоточною оптимізацією тривав 7ms;
– перший виклик сервісу з механізмом кешування тривав 427ms;
– багаторазовий виклик сервісу з механізмом кешування тривав 4ms.

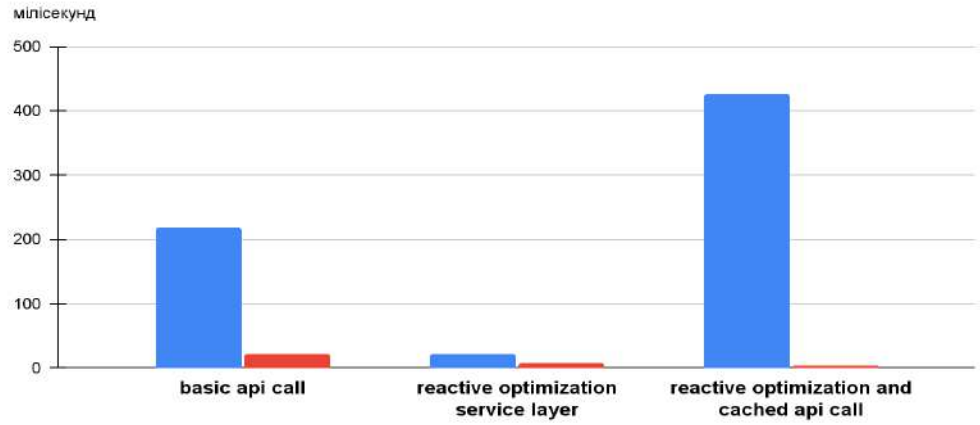


Рисунок 2 – Діаграма результатів експериментів першого та багаторазових викликів системи у декількох варіантах оптимізації

В залежності від вибору методу оптимізації можемо спостерігати зміну у швидкості обробки запитів.
Проведені експерименти дозволили зробити ряд висновків, зокрема, - незважаючи на те, що кешування запитів є затратна операція, але дозволяє зменшити час у 16 разів більше ніж звичайний GET HTTP запит.

Під час програмної реалізації фреймворка було використано реактивне програмування як засіб оптимізації сервісів. Тобто процес запиту замість звичайного поступового процесу розподіляється на декілька паралельних процесів на сервісі та виконується асинхронно. Цей спосіб дає приріст у швидкості обробки запиту у 5 разів навідміну від звичайного запиту.

Висновки. Оптимізація продуктивності роботи сервісів має першочергове значення для забезпечення ефективного та безперебійного обслуговування користувачів. Проведені експерименти дозволили сформулювати наступні рекомендації щодо напрямків підвищення продуктивності API:
- реалізація розумних стратегій кешування дозволяє тимчасово зберігати дані, що часто використовуються, тим самим знижуючи час очікування за рахунок мінімізації надлишкового пошуку та обробки даних;
- кешування є сенс використовувати лише у багато повторних запитах, у простих запитах краще використовувати звичайний запит до сервісу;
- використання методу PATCH для часткових оновлень може значно підвищити продуктивність;

Список інформаційних джерел

1. Bell, Michael. "Introduction to Service-Oriented Modeling". Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley & Sons. 2006, pp. 3. ISBN 978-0-470-14111-3.
2. K. Mani Chandy Event-Driven Applications: Costs, Benefits and Design Approaches, California Institute of Technology, 2006.

УДК 004.414

Сом Марія Олексіївна, *здобувачка вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник: Сперкач Майя Олегівна*, доцент кафедри інформаційних систем і технологій,*
КПІ ім. Ігоря Сікорського, Україна

ПОБУДОВА ОПТИМАЛЬНОГО НАВЧАЛЬНОГО ПЛАНУ З МЕТОЮ РОЗВИТКУ НАВИЧОК НАБОРУ ТЕКСТУ

Анотація. У роботі розглядаються побудова оптимального навчального плану для розвитку навичок сліпого та швидкісного набору тексту. Для вирішення даної задачі, на основі аналізу існуючих методів багатокритеріальної оптимізації (TOPSIS, VIKOR, AHP), було обрано метод AHP з певними модифікаціями. Наведено детальний опис обраного методу з переліком кроків.
КЛЮЧОВІ СЛОВА: навичка, навчальний план, набір тексту, TOPSIS, VIKOR, AHP.

Abstract. The work considers creating of an optimal educational plan for the development of blind and speed typing skills. To solve this problem, based on the analysis of existing methods of multi-criteria optimization (TOPSIS, VIKOR, AHP), the AHP method (with modifications) was chosen. A detailed description of the chosen method with a list of steps is provided.
KEY WORDS: skill, study plan, typing, TOPSIS, VIKOR, AHP.

Вступ. У сучасному світі технології супроводжують нас на кожному кроці, тому вміння ефективно взаємодіяти з ними стає все більш актуальним і необхідним для людей будь-якої професії. Завдяки навичкам швидкісного та сліпого набору тексту можна виконувати робочі завдання швидше, за рахунок чого збільшується й продуктивність роботи. Однак, на жаль, навчання швидкому та сліпому набору тексту є доволі складним і тривалим процесом, який потрібно правильно організувати, тому пошук оптимального навчального плану є актуальною проблемою кожного з нас.

Моя доповідь пропонує ефективний алгоритм побудови оптимального навчального плану, який буде корисним для людей, що прагнуть покращити навички швидкісного та сліпого набору тексту. Результати проведеного дослідження можуть бути використані на практиці шляхом впровадження в систему розумного навчання. Основою запропонованого алгоритму є метод багатокритеріальної оптимізації AHP, що забезпечує більш точний відбір уроків та підвищує ефективність навчального процесу.

Основна частина.

Змістовна постановка задачі.

Користувач прагне розвинути до майстерності s навичок набору тексту. Для досягнення поставленої мети він буде проходити l уроків, що передбачені його індивідуальним навчальним планом.

Поняття навички представляє собою впорядковану множину комбінацій літер, довжиною 2.

Мета – побудова оптимального навчального плану, тобто такого, що містить мінімальну кількість уроків, яка забезпечить досягнення майстерності.

Математична постановка задачі

s кількість навичок

l кількість уроків

$K = \{K_i\}$ множина критеріїв (навичок) $i = 1,\ldots, s$

$A = \{A_j\}$ множина альтернатив (уроків) $j = 1,\ldots, l$

$l \rightarrow min$

Мінімізація кількості уроків реалізується шляхом вибору найкращої на даний момент альтернативи на кожному кроці (після проходження кожного уроку).

Обґрунтування методу розв’язання.

TOPSIS – це метод багатокритеріальної оптимізації, що порівнює множину

альтернатив, нормалізуючи оцінки для кожного з критеріїв та обчислюючи геометричну відстань між кожною та ідеальною альтернативою (найкраща оцінка за кожним критерієм). [1]

VIKOR – це метод багатокритеріальної оптимізації, що знаходить компромісне ранжування та компромісне рішення, враховуючи критерії та їх ваги. Цей метод застосовує для ранжування багатокритеріальний індекс, що певним чином відображає «близькість» до ідеального рішення. [2]

AHP (Analytic Hierarchy Process, Saaty) – це метод багатокритеріальної оптимізації, основною задачею якого є формулювання та аналіз рішень.

Він застосовується для задач, де рішення базується на декількох критеріях, які мають різні ваги. AHP враховує відносні пріоритети альтернатив згідно з ієрархією критеріїв та альтернатив. [3]

Завдання полягає у побудові оптимального навчального плану, що вимагає врахування різних критеріїв та їх взаємозв'язку. AHP використовує парне порівняння альтернатив та критеріїв для визначення пріоритетів, що спрощує процес оцінки та вибору кращого варіанту серед доступних альтернатив.

Опис методу розв'язання.

Модифікуємо метод AHP шляхом інтеграції перевірки прийнятної переваги, у разі потреби також застосовуємо Баєсівський аналіз.

КРОК 1 Застосування AHP до проблеми оцінки та ранжування альтернатив.

КРОК 1.1 Декомпозиція проблеми передбачає побудову ієрархії взаємопов'язаних елементів, що описують проблему.

КРОК 1.2 Попарне порівняння здійснюється шляхом порівняння пар елементів у МПП(матриці прийняття рішень). Зазвичай використовується шкала від 1 до 9 для отримання вхідних даних.

МПП будується для кожного з нащадків і має наступну структуру:

	E_1	E_2	…	E_n
E_1	v_1/v_1	v_1/v_2	…	v_1/v_n
E_2	v_2/v_1	v_2/v_2	…	v_2/v_n
…	…	…	…	…
E_n	v_n/v_1	v_n/v_2	…	v_n/v_n

[3]

Задача передбачає побудову МПП для альтернатив та критеріїв.

Значення елементів МПП альтернатив розраховується як відношення добутку кількості входжень навички K_i в урок A_j та пріоритетету уроку для K_i до добутку кількості входжень навички K_i в урок A_k та пріоритетету уроку для K_i.

* i = 1…s, j = 1…*l*, k = 1…*l*.

Значення коефіцієнту пріоритету визначається наступним чином:

1) якщо $P(known)_i <= 50$, то значення коефіцієнту уроку, який включає:
символи – 3, слова – 2, текст – 1;
2) якщо $P(known)_i <= 75$, то значення коефіцієнту уроку, який включає:
символи – 1, слова – 2, текст – 1;
3) інакше – значення коефіцієнту уроку, який включає:
символи – 1, слова – 2, текст – 3.

*P(known) - ймовірність того, що учень уже володіє навичкою.

Значення елементів МПП критеріїв розраховується як відношення P(known) навички K_i до P(known) навички K_h.

* i = 1…s, h = 1…s.

КРОК 1.3 Перевірка узгодженості МПП.

- Характеристичне число знаходиться за формулою:
$$\lambda_{max} = \frac{x_i^{(m+1)}}{x_i^{(m)}}$$ [4]
- Використовується індекс узгодженості CI:
$CI = (\lambda max - k)/(k - 1)$ [4]
Для повністю узгодженої матриці λmax = k, а для неузгодженої завжди λmax > k, k (порядок системи) – кількість альтернатив/критеріїв.
- Визначається достатність ступеня узгодженості: $CR = CI/CIS$. [4]

CIS – середнє значення CR, розрахованих для значної кількості згенерованих випадковим чином МПП в фундаментальній шкалі.

k	1	2	3	4	5	6	7	8	9	10
CIS	0	0	0.52	0.89	1.11	1.25	1.35	1.4	1.45	1.49

Отриманий вектор відносних ваг альтернатив є прийнятним, у разі, якщо CR входить в діапазон, верхньою границею якого є значення 0,2. [4]

КРОК 1.4 Визначення відносних пріоритетів елементів з метою подальшого ранжування альтернатив.

- Визначення локальних пріоритетів елементів.

  Локальні пріоритети елементів – це коефіцієнти важливості альтернатив/ критеріїв для досягнення мети.

  Локальний пріоритет знаходиться, як нормалізоване середнє геометричне значення за відповідним рядком у МПП.
- Визначення глобальних пріоритетів альтернатив та критеріїв.

  $p_i = \sum_{j=1}^{m} \left(p_{ij}\omega_j\right)$. [4]

  p_{ij} - локальний пріоритет альтернативи і для j-го критерія,

  ω_j – локальний пріоритет j-го критерія.
- На основі глобальних пріоритетів відбувається ранжування альтернатив.

КРОК 2 Модифікація алгоритму шляхом інтеграції перевірки прийнятної переваги.

$$a'' - a' \geq \frac{a' - a^0}{n-1} \text{ [4]}$$

Де a^0 - найгірша перевага, a" – наступна найкраща альтернатива в ранжуванні після a', n - кількість альтернатив.

КРОК 3 Застосування Баєсівського аналізу. Якщо прийнятна перевага виконується, то обирається найкраща альтернатива, інакше додається модифікація – Баєсівський аналіз.

Для кожного критерію визначається такий підкритерій, як подібність до останніх 5 уроків. Далі розглядається тільки та множина найкращих альтернатив, що є задовільною для прийнятної переваги.

Локальні пріоритети критеріїв розраховуються за допомогою Баєсівського рівняння:

$$v\left(K_r/G\right) = \frac{v\left(K_r\right) * g\left(K_r\right)}{\sum_{i=1}^{s} v\left(K_i\right) * g\left(K_i\right)} \text{. [3]}$$

$v\left(K_r\right)$ – початковий локальний пріоритет критерію K_r.

$g\left(K_r\right)$ – рівень подібності за критерієм K_r.

$v\left(K_r/G\right)$ – результуючий локальний пріоритет критерію K_r.

$$g\left(K_r\right) = \frac{\sum_{i=1}^{5}\sum_{j=i+1}^{5}\left|n_{ir} - n_{jr}\right|}{10},$$

де r - номер навчички для якої проводиться розрахунок, n – кількість входжень навички r в урок i або j.

КРОК 4. Повторення кроків методу АНР, що ідуть після розрахунку локальних пріоритетів критеріїв, доки не буде виконуватись умова прийнятної переваги.

КРОК 5. Вибір найкращої за ранжуванням альтернативи.

Приклад застосування методу.

Початковий рівень навичок:
ab – 20%
ba – 30%
ac – 60%
ca – 90%
Статистичні дані можливих уроків:

i\s	ab	ba	ac	ca
1(символи)	1	2	1	1
2(слова)	1	1	1	1
3(текст)	1	1	2	1

МПП критеріїв:

K	ab	ba	ac	ca
ab	1	2/3	1/3	2/9
ba	3/2	1	½	1/3
ac	3	2	1	2/3
ca	9/2	3	3/2	1

ʎmax ≈ 4
CI = 0, CR = 0
МПП альтернатив за критерієм ab:

ab	1	2	3
1	1	1/2	1/2
2	2	1	1
3	2	1	1

ʎmax = 3
CI = 0, CR = 0

МПП альтернатив за критерієм ac:

ac	1	2	3
1	1	1/2	1/2
2	2	1	1
3	2	1	1

ʎmax = 3
CI = 0, CR = 0
МПП альтернатив за критерієм ba:

ba	1	2	3
1	1	3	6
2	1/3	1	2
3	1/6	1/2	1

ʎmax = 3
CI = 0, CR = 0
МПП альтернатив за критерієм ca:

ca	1	2	3
1	1	1/2	1/3
2	2	1	2/3
3	3	3/2	1

ʎmax = 3
CI = 0, CR = 0
Локальні пріоритети:

k	ab	ba	ac	ca
0.47 → 0.1	1.65 → 0.5	2.62 → 0.67	0.63 → 0.2	0.55 → 0.17
0.71 → 0.15	1.1 → 0.33	0.87 → 0.22	1.26 → 0.4	1.1 → 0.33
1.41 → 0.3	0.55 → 0.17	0.44 → 0.11	1.26 → 0.4	1.65 → 0.5
2.12 → 0.45				

Результуюче ранжування (глобальні пріоритети): [0.29; 0.33; 0.38].

Умова прийнятної переваги виконується:
0.05 = 0.38 – 0.33 ≥ 0.045 = (0.38 – 0.29)/(3-1)
Отже, 3-ій типу уроку, тобто текстовий є найбільш оптимальним в даному випадку.

Висновки. У доповіді досліджено проблему побудови оптимального навчального плану для розвитку навичок швидкісного та сліпого набору тексту, сформульовано змістовну постановку задачі.

З метою досягнення максимальної ефективності було застосовано метод багатокритеріальної оптимізації AHP, адже він використовує парне порівняння альтернатив та критеріїв для визначення пріоритетів, завдяки чому процес оцінки уроків та вибір найкращого значно спрощується. Даний метод доцільно використовувати як складову розумної системи розвитку навичок, що допоможе підібрати найбільш ефективні уроки та методи досягнення максимального результату.

Результати проведеного дослідження можуть бути використані щоб збільшити ефективність навчання швидкісного та сліпого набору тексту, а також допомогти учням швидше досягти майстерності.

Список інформаційних джерел

1. Subrata C. TOPSIS and Modified TOPSIS: A comparative analysis. Посилання:https://www.sciencedirect.com/science/article/pii/S277266222100014X
2. Sayadi M. K., Heydari M., Shahanaghi K. Extension of VIKOR method for decision making problem with interval numbers. Applied Mathematical Modelling. 2009. Volume 33. P. 2257–2262. Посилання:https://www.sciencedirect.com/science/article/pii/S0307904X08001558
3. Mimović P., Stanković J., Milić V. J. Decision-making under uncertainty – the integrated approach of the AHP and Bayesian analysis. Economic Research-Ekonomska Istraživanja. 2015. Volume 28. P. 868–878.
Посилання: https://www.tandfonline.com/doi/full/10.1080/1331677X.2015.1092309
4. Теорія прийняття рішень. Навч. посіб. для студ. спеціальності 126 «Інформаційні системи та технології» та спеціальності 121 «Інженерія програмного забезпечення» / КПІ ім. Ігоря Сікорського; уклад.: О.С. Жураковська. Київ: КПІ ім. Ігоря Сікорського, 2020. – 99 с. Посилання:https://www.tandfonline.com/doi/full/10.1080/1331677X.2015.1092309

УДК 004.414

Куник Неля Вікторівна, *здобувачка вищої освіти*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник: Сперкач Майя Олегівна*, доцент кафедри інформаційних систем і технологій,*
КПІ ім. Ігоря Сікорського, Україна

ВІДСТЕЖЕННЯ ЗНАНЬ У СИСТЕМІ ДЛЯ РОЗВИТКУ НАВИЧОК ШВИДКІСНОГО ТА СЛІПОГО НАБОРУ ТЕКСТУ

Анотація. У доповіді розглянута проблема відстеження знань під час навчання швидкісному та сліпому набору тексту. Наведено огляд актуальних методів відстеження знань, їх порівняння в контексті вирішення даної задачі, запропоновано алгоритм відстеження рівня знань користувача, а також прогнозування його прогресу, який передбачає застосування алгоритму BKT та моделі Раша.
КЛЮЧОВІ СЛОВА: швидкісний набір, відстеження знань, прогнозування, BKT, IRT.

Abstract. The work examines the problem of tracking knowledge during learning speed and blind typing. An overview of current methods of knowledge tracking is given, their comparison in the context of solving this problem, an algorithm for tracking the user's knowledge level, as well as forecasting its progress, which involves the use of the BKT algorithm and the Rasch model, is proposed.
KEY WORDS: speed typing, knowledge tracking, prediction, BKT, IRT.

Вступ. В добу розвитку інформаційних технологій ключовим фактором на шляху до успіху є вміння їх ефективного використання. Нині швидкість і продуктивність є одними з основних покажчиків якісної роботи, тому знання навичок швидкісного та сліпого набору тексту стає з кожним роком все більш важливим. Однак, процес здобуття цих навичок є досить нетривіальним та потребує тривалого тренування й багато практики.

Можливим способом розвитку навичок набору тексту є використання спеціальної розумної системи навчання. Розумна система не просто надає користувачам можливість тренуватися, важливим аспектом роботи такої системи є відстеження прогресу та рівня навичок користувачів. Тому моя доповідь пропонує алгоритм відстеження знань для задачі розвитку навичок швидкісного та сліпого набору тексту, який надалі може використовуватись як складова системи розумного навчання. Основою запропонованого алгоритму є метод BKT(Bayesian Knowledge Tracing) а також модель Раша IRT(Item Response Theory).

Метою доповіді є демонстрація алгоритму відстеження знань користувачів у системі для розвитку навичок швидкісного та сліпого набору тексту, що допоможе забезпечити високий рівень персоналізації навчання та спросити визначення поточного рівня володіння навичками.

Основна частина.

Змістовна постановка задачі

Користувач прагне розвинути s навичок у швидкому наборі тексту на клавіатурі, проходячи уроки. Навичка являє собою впорядковану множину символів, потужністю 2.

Користувач досягне майстерності, якщо на момент останнього уроку l для кожної навички частка міскліків w становитиме 5/100 (або менше), а час набору символу $t^{\text{сим}}$ – 1/600 хвилини (або менше).

Таким чином мета користувача – досягти майстерності, а отже наведених вище характеристик для кожної з навичок.

Математична постановка задачі

s кількість навичок
l номер останнього уроку

w_{il} частка міскліків навички i під час проходження уроку l; i = 1,…, s

$t_{il}^{\text{сим}}$ середній час набору 1 символа навички i під час проходження уроку l (хвилини); i = 1,…, s

n_{il} кількість входжень навички i в урок l; i = 1,…, s

m_{il} кількість випадків неправильного введення навички i під час проходження уроку l; i = 1,…, s

t_{il} максимальний час набору навички I під час проходження уроку l (хвилини); i = 1,…, s

$P(willlearn)_{il}$ ймовірність того, що учень засвоїв навичку i під час проходження уроку l (хвилини); i = 1,…, s

Для $\forall i \in [1; s]$:

$$t_{il}^{\text{сим}} \to \frac{1}{600}\ w_{il} \to \frac{5}{100}$$

$$t_{il}^{\text{сим}} = \frac{t_{il}}{k_i} \Rightarrow \frac{t_{il}}{k_i} \to \frac{1}{600}$$

$$w_{il} = \frac{m_{il}}{n_{il}} \Rightarrow \frac{m_{il}}{n_{il}} \to \frac{5}{100}$$

$$P(willlearn)_{il} = \frac{\left(1-\frac{m_{il}}{n_{il}}\right)*1}{t_{il}*600} = P(willlearn)_{il} \to 0.95$$

Обґрунтування методу розв'язання

Bayesian Knowledge Tracing (BKT) - це алгоритм, що використовується в Intelligent Tutoring Systems. Він дозволяє нам зробити висновок про поточний стан знань учня, щоб передбачити чи він засвоїв навичку. [1]

BKT припускає, що знання учня представлені як набір змінних від 0 до 1, по одній на навичку, де навичка або засвоєна учнем, або ні.

Моделювання процесу навчання за допомогою BKT зазвичай відбувається на основі статистичних даних, таких як результати тестів. Алгоритм визначає ймовірність того, що студент володіє конкретною навичкою на основі його попереднього досвіду та результатів пройдених тестів.

Основна задача алгоритму BKT - спрогнозувати рівень успішності студентів та визначити, які знання вони засвоїли, а які ще потребують удосконалення.

Також до розв'язання даної задачі може бути застосовано Deep Knowledge Tracing (DKT) алгоритм. Основна ідея алгоритму DKT полягає в моделюванні процесу навчання студента та його здатності засвоювати знання з кожним новим завданням. Використовуючи історію відповідей студента на попередні завдання, DKT робить прогноз щодо його рівня знань наступного разу, коли він зустріне нове завдання.

DKT застосовує рекурентну модель нейронної мережі для оцінки засвоєння учнями навичок. Він передбачає виконання послідовності вправ, які є набором запитань, що відносяться до різних концепцій. Кожна відповідь є послідовністю вхідних даних, що включають ідентифікатор завдання та відповідь студента на завдання. [2] Тому DKT може оцінити рівень знання студентами кількох навичок одночасно.

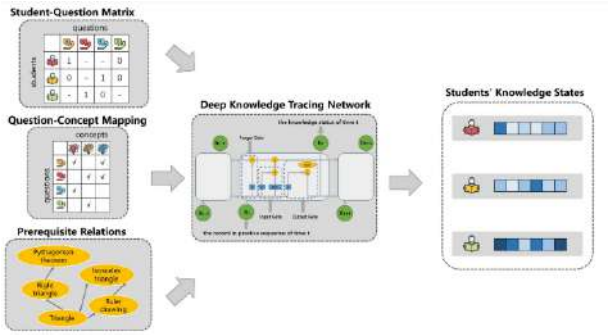


Рисунок 1. Ілюстрація проблеми відстеження знань[3]

Таким чином, модель навчається на історії відповідей студента і на основі поточного рівня знань робить передбачення щодо можливості правильної відповіді.

BKT це класичний алгоритм відстеження знань, який легко застосувати. Він є простішим за DKT, а також потребує менше параметрів та обчислювальних ресурсів. Даний алгоритм добре підходить для випадків, коли знання, що здобуваються учнями, представлені невеликою кількістю окремих навичок (як у нашому випадку, з дволітерними комбінаціями).

На відміну від вище згаданого алгоритму, DKT є більш потужним, проте він використовується у випадках, де потрібно одночасно відстежувати велику кількість навичок чи концепцій. Наприклад, він застосовується при навчанні грі на музичному інструменті або ж при вивченні мови програмування. Також DKT є складнішим у реалізації та інтерпретації результатів.

Таким чином, для студентів, які навчаються сліпому набору тексту за допомогою дволітерних комбінацій, алгоритм BKT більше підходить, аніж DKT.

Опис методу розв'язання

КРОК 1 Спочатку учень проходить три тестові уроки, результати кожного з яких обробляються за допомогою моделі Раша – однопараметрична модель IRT (Item Response Theory).

КРОК 1.1 Розрахунок параметру P(will learn) на основі статистичних даних навичок пройденого уроку за формулою:

$$P(willlearn)_{il} = \frac{\left(1-\frac{m_{il}}{n_{il}}\right)*1}{t_{il}*600}.$$

КРОК 1.2 Визначення значення параметра P(known).

У моделі Раша підсумковий бал є результатом взаємодії прихованих параметрів, а саме початкового рівня підготовки учнів і складності завдань, що були розглянуті. [4]

$$P(willlearn)_{il} = \frac{e^{1,7*\left(P(known)-D_k\right)}}{1+e^{1,7*\left(P(known)-D_k\right)}}, [6, 7]$$

де:

— $P(\text{will learn})_i$ – вірогідність того, що учень з рівнем навички P(known) відповість правильно на завдання зі складністю D_k,

— i – номер навички,

— k – номеру завдання.

Примітка:

— $P(\text{will learn})_i$, D_k, i = 1,…, s, k =1,…, 3 – відомі параметри,

— P(known) визначається методом максимальної правдоподібності.

Параметр D_k дорівнює:

— 3, якщо тестовий урок складається з невпорядкованого набору навичок

— 2, якщо тестовий урок складається з складних довгих слів

— 1, якщо тестовий урок складається з простих коротких слів

Примітка: Кожний з тестових уроків включає всі навички по 1 разу.

КРОК 2 Результати уроків індивідуального навчального плану обробляються за допомогою моделі BKT.

Існує чотири параметри, задіяні в BKT (кожен має значення від 0 до 1 включно):

— P(known): ймовірність того, що учень уже володіє навичкою.

— P(will learn): ймовірність того, що студент засвоїть навичку під час поточної практики.

— P(slip): ймовірність того, що учень відповість неправильно, незважаючи на знання навички.

— P(guess): ймовірність того, що учень відповість правильно, незважаючи на те, що не володіє навичкою.

[1]

КРОК 2.1 Визначення параметру P(guess).

P(guess) приймає значення в діапазоні 0,01 – 0,2, відповідно до рівня набутої навички перед проходження даного уроку. При цьому, 0,01 відповідає 100%-ому володінню навичкою, тоді як 0,2 – нульовому.

КРОК 2.2 Визначення параметру P(slip).

P(slip) приймає значення в діапазоні 0,01 – 0,2, відповідно до рівня набутої навички перед проходження даного уроку. При цьому, 0,01 відповідає нульовому володінню навичкою, тоді як 0,2 – 100%-ому.

КРОК 2.3 Визначення параметру P(will learn) на основі статистичних даних навичок пройденого уроку за формулою:

$$P(will\ learn)_{il} = \frac{\left(1-\frac{m_{il}}{n_{il}}\right)*1}{t_{il}*600}.$$

КРОК 2.4 Якщо P(will learn) ≥ 0.95 для кожної навички, ми говоримо, що учень досяг найкращих результатів.

КРОК 2.5 Визначення параметру P(learned), якщо учень не досяг майстерності.

P(learned) - ймовірність того, що учень засвоїв навичку, над якою він працював, використовуючи значення попередніх параметрів.

КРОК 2.5.1 Обчислення умовної ймовірності того, що студент засвоїв навичку раніше (у момент часу n-1), виходячи з того, чи правильно він відповів на поточне запитання (в момент часу n).

$$P(learned_{n-1} \mid correct_n) =$$

$$= \frac{P(known_{n-1}) * (1 - P(slip))}{P(known_{n-1}) * \left(1 - P(slip)\right) + \left(1 - P(known_{n-1})\right) * P(guess)}$$

[5]

$$P(learned_{n-1} \mid incorrect_n) =$$

$$= \frac{P(known_{n-1}) * P(slip)}{P(known_{n-1}) * P(slip) + (1 - P(known_{n-1})) * (1 - P(guess))}$$

[5]

$P(learned_{n-1}|answer_n)$ розраховується як середнє значення серед усіх $P(learned_{n-1}|correct_n)$ та $P(learned_{n-1}|incorrect_n)$.

КРОК 2.5.2 Розрахунок умовної ймовірності того, що студент засвоїв навичку зараз (у момент часу n), використовуючи результат першого обчислення.

$$P(learned_n \mid answer_n) =$$

$$= P(learned_{n-1} \mid answer_n) + (1 - P(learned_{n-1} \mid answer_n)) * P(will\ learn)$$

[5]

КРОК 2.6 Присвоїти P(known) значення P(learned)

Приклад застосування методу для нетестового уроку.

	ab	ba	ac	ca
m	0	1	1	0
c	7	8	5	4
t	0.04	0.02	0.01	0.002

Таблиця 1. Статистичні дані отримані після проходження уроку

P(known)	0.2	0.3	0.6	0.9
P(guess)	0.16	0.14	0.08	0.02
P(slip)	0.04	0.06	0.12	0.18
P(will learn)	0.04	0.07	0.13	0.08

Таблиця 2. Результати виконання кроків 2.1-2.4

Розрахунки для кроку 2.3:

P(will learn):

1) $\frac{1-\frac{0}{7}}{0.04*600} = \frac{1}{24}$ (ab)

2) $\frac{1-\frac{1}{8}}{0.02*600} = \frac{7}{96}$ (ba)

3) $\frac{1-\frac{1}{5}}{0.01*600} = \frac{2}{15}$ (ac)

4) $\frac{1-\frac{0}{4}}{0.02*600} = \frac{1}{12}$ (ca)

Розрахунки для кроку 2.5:

$P(learned_{n-1}|correct_n)$:

$\frac{0.2*(1-0.04)}{0.2(1-0.004)+(1-0.2)*0.16} = 0.6$

$\frac{0.3*(1-0.06)}{0.3(1-0.006)+(1-0.3)*0.14} = 0.7$

$\frac{0.6*(1-0.12)}{0.6(1-0.12)+(1-0.6)*0.08} = 0.9$

$\frac{0.9*(1-0.18)}{0.9(1-0.18)+(1-0.9)*0.02} = 0.99$

P(learnedn-1|incorrectn):

$\frac{0.3*0.06}{0.3*0.06+(1-0.3)*(1-0.14)} = 0.03$

$\frac{0.6*0.12}{0.6*0.12+(1-0.6)*(1-0.08)} = 0.16$

$P(learned_{n-1}|answer_n)$:

0.6

$\frac{0.7*7+0.03*1}{8} = 0.62$

$\frac{0.9*4+0.16*1}{5} = 0.75$

0.99

$P(learned_n|answer_n)$:

1) 0.6 + (1 - 0.6)*0.04 = 0.616
2) 0.62 + (1 – 0.62)*0.07 = 0.65
3) 0.75 + (1 – 0.75)*0.13 = 0.78
4) 0.99 + (1 – 0.99)*0.08 = 0.99

Таким чином ми відстежили розвиток кожної з 4х навичок, а також дізналися, що навичка "ca" не потребує подальшого розвитку, адже користувач вже досяг майстерності.

Висновки. У доповіді було досліджено проблему відстеження рівня знань на прикладі задачі швидкісного та сліпого набору тексту.

Для вирішення даної проблеми було застосовано поєднання алгоритму BKT та моделі Раша, оскільки BKT використовується для відстеження знань, які представлені невеликою кількістю окремих навичок, а за допомогою моделі Раша та тестових завдань відбувається розрахунок параметру, що являє собою рівень знань користувача на момент початку його взаємодії з системою розумного навчання. Дане рішення може покращити якість навчання, оскільки дозволяє системі точно відстежувати рівень засвоєння навичок й надалі адаптувати уроки до потреб кожного з користувачів.

Також були визначені умови досягнення майстерності у навичках, що дає можливість легко визначити момент зупинки розвитку певної навички.

Отримані результати можуть бути корисні для розробки більш розумних систем розвитку навичок швидкісного та сліпого набору тексту, які б дозволяли користувачам ефективніше досягати майстерності.

Список інформаційних джерел

1. Yudelson M. V., Koedinger K. R., Gordon G. J. Individualized Bayesian Knowledge Tracing Models. URL:
https://www.cs.cmu.edu/~ggordon/yudelson-koedinger-gordon-individualized-bayesian-knowledge-tracing.pdf
2. Deep Knowledge Tracing / C. Piech et al.
URL: https://stanford.edu/~cpiech/bio/papers/deepKnowledgeTracing.pdf
3. Prerequisite-Driven Deep Knowledge Tracing / P. Chen et al. International Conference on Data Mining, 2018. URL: https://aic-fe.bnu.edu.cn/docs/20190109161907836283.pdf
4. Wiberg M. Classical test theory vs. item response theory. EM No 50. 2004. URL:
https://dokumen.tips/documents/classical-test-theory-vs-item-response-test-theory-vs-item-response-theory-an.html

УДК 004.42

Олексієвець Олександр Володимирович, *магістрант 2 курсу*
КПІ ім. Ігоря Сікорського, Україна
Науковий керівник: Демчишин Анатолій Анатолійович, *кандидат технічних наук, доцент*
КПІ ім. Ігоря Сікорського, Україна

ПІДВИЩЕННЯ РОЗДІЛЬНОЇ ЗДАТНОСТІ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

КЛЮЧОВІ СЛОВА: підвищення роздільної здатності зображень, зменшення ваги, випадання нейронів, нормалізація пакетів.

Зображення є одним із найбільш поширених та важливих типів даних в сучасному світі. Вони використовуються в багатьох галузях, таких як медицина, промисловість, розваги, наука та ін. Одним з найважливіших параметрів зображення є його роздільна здатність, яка визначається кількістю пікселів на одиницю довжини. Підвищення роздільної здатності зображень може значно покращити їх якість та деталізацію, що є важливим для багатьох застосувань. Нейронні мережі в наш час є доволі потужним інструментом для роботи з зображеннями та обробки інших типів даних. Проблема підвищення роздільної здатності зображень і використання для цього нейронних мереж є актуальною з кількох причин:

1. Покращення якості зображень. Висока роздільна здатність зображень є важливою для багатьох застосувань, таких як медична діагностика, візуальне сприйняття, безпека та нагляд. Збільшення роздільної здатності зображень дозволяє отримати більш детальну інформацію, що може допомогти у поліпшенні точності аналізу, діагностики та розпізнавання об'єктів.
2. Розвиток області штучного інтелекту. Нейронні мережі є потужним інструментом у сфері штучного інтелекту, і забезпечення їх здатності до обробки зображень з високою роздільною здатністю важливо для багатьох завдань, таких як комп'ютерне зорове сприйняття, розпізнавання образів та глибинне навчання. Завдяки розвитку нейронних мереж із здатністю до підвищення роздільної здатності, можна досягти значного прогресу у багатьох галузях.
3. Зростання обчислювальних можливостей. Завдяки швидкому розвитку обчислювальних технологій, таких як графічні процесори (GPU) та тензорні прискорювачі, стало можливим ефективно виконувати обчислення для обробки великих зображень з високою роздільною здатністю. Це відкриває нові можливості для застосування нейронних мереж у завданнях підвищення роздільної здатності зображень.

Існує багато архітектур нейронних мереж, які можуть вирішити цю проблему, проте вибір структури мережі та методів її оптимізації, що позитивно вплинули б на навчання мережі та дозволили уникнути проблеми перенавчання, залишається викликом.

З метою вирішення цього завдання було створено програмний застосунок, який дозволяє підвищувати роздільну здатність зображення, на основі навченої нейромережі для підвищення роздільної здатності зображень (SRCNN). Для оптимізації навчання мережі розглянуто наступні методи регуляризації мережі: зменшення ваги, випадання нейронів, нормалізація пакетів.

Зменшення ваги (weight decay) це техніка регуляризації шляхом додавання певного штрафу, найчастіше цим штрафом виступає Евклідова норма [1] (L2 norm) вагових коефіцієнтів, до функції втрат:

$$loss = loss + weight\ decay\ parameter * L2\ norm\ of\ the\ weights,$$

де N - кількість прикладів у тренувальному датасеті, y - цільові значення, f(x) - прогнозовані значення, w - вага моделі.

Використання цієї техніки [1]:

1. дозволяє уникнути перенавчання мережі;
2. зменшує вплив великих значень ваг на модель;
3. забезпечує більш точні прогнози на нових даних;

4. дозволяє зберегти значення вагових коефіцієнтів малими та уникнути вибуху градієнта: оскільки Евклідова норма додається до функції втрат, то на кожній ітерації мережа намагатиметься оптимізувати вагові коефіцієнти невідривно від самої функції втрат, це дозволить зберігати якомога менші значення вагових коефіцієнтів, і таким чином уникнути виникнення вибуху градієнту.

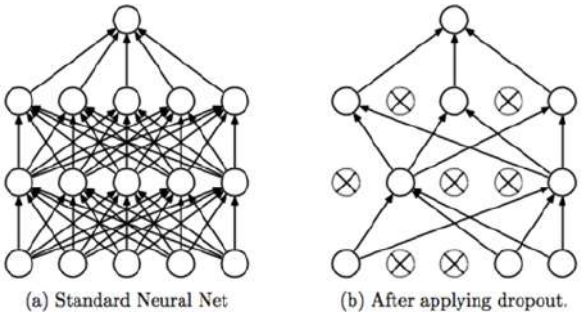


Рисунок 1 - Приклад використання техніки випадання нейронів на одній з ітерацій навчання мережі

Випадання нейронів (dropout) є одним із методів регуляризації нейронних мереж, який допомагає уникнути перенавчання та забезпечує більш точні прогнози на нових даних. Він зменшує взаємозв'язок між шарами нейронної мережі, що призводить до випадкового зменшення числа параметрів моделі та збільшення різноманітності між моделями, що в результаті допомагає уникнути перенавчання.

Під “випаданням” мають на увазі, що ці одиниці не беруться до уваги при проходженні даних через них вперед або назад [2] (рисунок 1).

Нормалізація пакетів (batch normalization, далі BN) є одним із методів регуляризації нейронних мереж, який забезпечує стабільність та швидкість навчання [3]. Він зменшує взаємозв'язок між шарами нейронної мережі та забезпечує сталу нормалізацію ваг та зміщень.

Складається цей метод з нормалізації векторів активації прихованих шарів нейронної мережі за допомогою використання середнього значення (mean) та дисперсії поточного batch’у. Нормалізація використовується перед або після використання нелінійної функції.

BN забезпечує нормалізацію вихідних значень по всій групі зображень (batch) і зменшує кореляцію між різними каналами зображення. Це зменшує вплив варіацій між зображеннями та допомагає моделі швидше зберігати та відтворювати деталі зображення [4].

Висновки. Таким чином, застосування BN допомагає збільшити роздільну здатність зображень у нейронних мережах, знижує вплив шуму та варіацій між зображеннями, і забезпечує більш швидке та стабільне навчання.

Список інформаційних джерел

1. On the Periodic Behavior of Neural Network Training with Batch Normalization and Weight Decay / E. Lobacheva et al. URL: https://arxiv.org/pdf/2106.15739.pdf.
2. Brownlee J. A gentle introduction to dropout for regularizing deep neural networks - machinelearningmastery.com. MachineLearningMastery.com. URL: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/
3. Ioffe S., Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. URL: http://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/43442.pdf.
4. Adaptive Cross Batch Normalization for Metric Learning / T. Ajanthan et al. URL: https://arxiv.org/pdf/2303.17127.pdf

Наукове видання

IV МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ МОЛОДИХ ВЧЕНИХ ТА СТУДЕНТІВ «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ І ПЕРЕДОВІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ» (SOFTTECH-2023)

9-11 травня 2023 року

Тези доповідей

Електронне видання.
Формат А4. Умовних друкарських аркушів 6.