Оргкомітет конференції

Голова організаційного комітету: О.А. Павлов – завідувач кафедри АСОІУ, д.т.н., професор.

Члени організаційного комітету:

П.І.П.	Посада
О.А. Павлов	Завідувач кафедри АСОІУ, професор
І.П. Муха	Доцент кафедри АСОІУ
О.В. Гавриленко	Доцент кафедри АСОІУ
І.В. Баклан	Доцент кафедри АСОІУ
К.І. Ліщук	Доцент кафедри АСОІУ
Ю.О. Олійник	Старший викладач кафедри АСОІУ

Члени програмного комітету:

П.І.П.	Посада
В.М. Томашевський	Професор кафедри АСОІУ
І.В. Стеценко	Професор кафедри АСОІУ
В.І. Литвиненко	Професор ХНТУ
Г.В. Рудакова	Професор ХНТУ
О.Г. Жданова	Доцент кафедри АСОІУ
Т.В. Ковалюк	Доцент кафедри АСОІУ
Е.В. Жаріков	Доцент кафедри АСОІУ
Ю.М. Селін	Доцент кафедри АСОІУ
О.С. Жураковська	Доцент кафедри АСОІУ
Т.О. Тєлишева	Доцент кафедри АСОІУ
В.Д. Попенко	Доцент кафедри АСОІУ
М.О. Сперкач	Доцент кафедри АСОІУ
М.І. Цюцюра	Доцент кафедри АСОІУ

ІІ Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління – ІСТУ-2019». Секція кафедри автоматизованих систем обробки інформації і управління. Матеріали конференції. – Київ. – 2019.

16 – 17 травня 2019р. – 130 с.

У збірник включені тези доповідей, які були представлені на конференції “Інформаційні системи та технології управління – ІСТУ-2019” в секції кафедри автоматизованих систем обробки інформації і управління. В доповідях розглянуті наукові та методичні питання щодо сучасних аспектів інформатики та обчислювальної техніки.

Редакційна колегія:

Гавриленко О.В., к.ф.-м.н., кафедра АСОІУ КПІ ім. Ігоря Сікорського, Муравйова І. М., інженер ІІ категорії кафедри АСОІУ

Дизайн титульної сторінки: Майєр З.О.

Зміст

УДК 004.912

ЛОТОЦЬКА Ю.В.
ХАЛУС О. А.

АНАЛІЗ МЕТОДІВ АВТОМАТИЧНОГО РЕФЕРУВАННЯ ТЕКСТУ

У даній статті проаналізовані основні підходи автоматичного реферування тексту. Представлений огляд різних процесів автоматичного реферування тексту і описано ефективність та недоліки різних методів.
КЛЮЧОВІ СЛОВА: автоматичне реферування, екстрактне реферування, абстрактне реферування

In this article, the main approaches to automatic text abstraction are analyzed. An overview of the various processes of the automatic text summarization is presented and the effectiveness and disadvantages of various methods are described.
KEYWORDS: automatic summarization, extractive summarization, abstractive summarization

1. Вступ

В останні роки спостерігається стрімке зростання обсягів текстової інформації що призводить до інформаційного перевантаження. В таких умовах зростає актуальність застосування методів автоматичного реферування, які допомагають користувачеві ефективно обробляти великі об’єми інформації.

Системи реферування текстів сприяють скороченню вмісту текстового документа і отриманню стислого реферату, в якому зберігається основний зміст оригіналу. Це полегшує процес пошуку та виявлення корисної інформації користувачем та дозволяє оперативно визначити необхідність звернення до першоджерела.

2. Класифікація методів автоматичного реферування

Виділяють два основних підходи до автоматичного реферування текстових документів: екстракція(sentence extraction, вилучення речень) та абстракція(abstractive summarization, генерування)[1].

При використанні методу екстракції із тексту виділяють найбільш важливі абзаци, речення. Але при цьому данні фрагменти не опрацьовуються, а вилучаються із документу в тому вигляді та порядку, що й і у оригінальному тексті.

Генеруючі методи реферування тексту засновані на лінгвістичних правилах обробки природної мови або методах штучного інтелекту. Генеруючі методи здатні створювати новий текст, не представлений явно в тексті початкового документа.

Детальніше розглянемо методи екстракції [2].

В роботі [3] розглядається поєднання різних ознак важливості речення таких, як поверхневі ознаки (розташування речень) та змістовні (частота слів). Було розглянуто два алгоритми: алгоритм навчання з учителем і алгоритм часткового навчання.

В методах основаних на теорії графів текст подається у вигляді графа, вузли якого представляють фрагменти тексту(слова, речення, абзаци), а ребра - позначають зв’язки між вузлами, наприклад семантичні зв’язки.

Окремо варто виділити метод фрактального реферування, запропонований в роботі [4]. Більшість статистичних методів реферування розглядають вихідний документ як лінійну послідовність речень, але ігнорують його структуру. Метод фрактального реферування грунтується на досить популярній в даний час математичній теорії фракталів.

Фрактальний принцип самоподібності передбачає нескінченне дроблення набору об'єктів зі збереженням їх властивостей. Реферування вихідного документа проводиться шляхом багаторазового використання перетворень стискання в теорії фракталів. За аналогією з фрактальної геометрією великий документ має ієрархічну структуру з декількома рівнями: глави, розділи, підрозділи, пункти, речення, поняття і слова. Хоча документ не є істинною математичною моделлю фрактального об'єкта, так як не може розглядатися в нескінченному рівні абстракції, структуру документа можна розглядати як псевдо-фрактальную з кінцевої рекурсією.

Класичний метод

Метод автоматичного реферування Едмундсона поєднує статистичний метод Г. Луна із позиційним та індикаторним методами.

Даний метод характеризує модель лінійних вагових коефіцієнтів[5]:

$$Weight(U) = Location(U) + Cuephrase(U) + Statterm(U) + Addterm(U)$$

Статистична важливість текстового блоку $Statterm(U)$ обчислюється як нормована по довжині блоку сума ваг слів або словосполучень, що до нього входить.

Вага $Location(U)$ визначається розташуванням блоку в початковому тексті і залежить від того, де з'являється даний фрагмент: на початку, в середині або в кінці, а також чи використовується він в ключових розділах тексту, наприклад, в заголовку або висновках.

Ключовими фразами(cue phrases) є лексичні конструкції-маркери, такі як «підсумовуюче вище сказане», «у даній статті», «за результатами аналізу» та інше. Стоп слова вилучаються із оригінального документа.

Байєсівський класифікатор

Наївний Байєсівський класифікатор є контрольованим методом навчання. Під час реферування тексту класифікація Байєса розглядає вибір речення як проблему класифікації. За цією класифікацією кожне речення ставиться в двійковий клас, щоб визначити, чи буде воно включене до реферату або ж ні. В цьому методі використовуються: частота слів, великі слова, довжина речення, позиція в абзаці і структура фрази. Розглядаючи k особливості та використовуючи правило Баєса, ймовірність того, що речення s буде включене у реферат S, визначається за формулою:

$$P(s \in S|F_1, F_{2\ldots}F_i) = \frac{\prod_{j=1}^{k} P\left(F_j \middle| s \in S\right) P(s \in S)}{\prod_{j=1}^{k} P(F_j)}$$

де $P(s \in S)$ є константою, $P\left(F_j \middle| s \in S\right)$ та $P(F_j)$ оцінюються з навчальних даних;

$F_1, F_{2\ldots}F_i$ - речення для класифікації, а S є рефератом, що генерується. Потім кожне речення оцінюється відповідно до цієї формули і ранжується для реферування.

Term Frequency - Inverse Document Frequency

TF-IDF - статистичний показник, що використовується для оцінки важливості слів у контексті документа, що є частиною колекції документів чи корпусу. Вага(значимість) слова пропорційна кількості вживань цього слова у документі, і обернено пропорційна частоті вживання слова у інших документах колекції[7].

TF(term frequency - частота слова) - це відношення числа входжень обраного слова до загальної кількості слів документа. Таким чином, оцінюється важливість слова t_i в межах обраного документа.

$$TF = \frac{n_i}{\sum_k n_k}$$

де n_i є число входжень слова в документ, а в знаменнику — загальна кількість слів в документі.

IDF (*inverse document frequency* — обернена частота документа) — інверсія частоти, з якою слово зустрічається в документах колекції. Використання IDF зменшує вагу широковживаних слів.

$$IDF = log\frac{|D|}{|(d]t, \in d)|},$$

де |D| - кількість документів колекції;

$|(d]t, \in d)|$ - кількість документів, в яких зустрічається слово t_i (коли $ni \neq 0$).

Вибір основи логарифму у формулі не має значення, адже зміна основи призведе до зміни ваги кожного слова на постійний множник, тобто вагове співвідношення залишиться незмінним.

Показник TF-IDF це добуток двох множників: TF та IDF.

Більшу вагу TF-IDF отримають слова з високою частотою появи в межах документа та низькою частотою вживання в інших документах колекції.

Метод штучної нейронної мережі

Штучна нейронна мережа - це обчислювальна модель, що використовується в інформатиці та інших областях досліджень для вирішення завдань на основі підходів до машинного навчання. Кайхах використовував штучні нейронні мережі як метод екстракції речень для узагальнення новинних статей. Метод поділяється на три етапи: нейромережеве навчання, функціональне злиття і вибір речення.

Етап навчання визначає типи речень, які повинні бути представлені в рефераті документа. Людина робить це і система вивчає схему реферування речень. Після навчання штучної нейронної мережі слід визначити співвідношення між ознаками. Під час навчання машини розглядаються наступні сім ознак:

1. Параграф після назви документу
2. Розташування абзацу в документі
3. Розташування речення в абзаці
4. Перше речення параграфу
5. Довжина речення
6. Кількість тематичних слів у реченні
7. Кількість заголовків у реченні.

Цей крок складається з двох етапів: 1) видалення незвичайних ознак і 2) усунення ефектів загальних ознак. Тому цей крок узагальнює важливі особливості, які повинні існувати в підсумкових реченнях. Після навчання та узагальнення мережі ця система може використовуватися для вибору важливих речень для реферування текстового документа.

Латентно-семантичний аналіз

Метод латентно-семантичного аналізу (ЛСА) дозволяє виявляти значення слів з урахуванням контексту їх використання шляхом обробки великого обсягу текстів.

Модель представлення тексту, що використовується в латентно-семантичному аналізі, багато в чому схожа з сприйняттям тексту людиною. Наприклад, за допомогою цього методу можна оцінити текст на відповідність заданій темі.

В якості вихідної інформації використовується терм-документа матриця.

Терм-документа матриця - це математична матриця, що описує частоту термінів, які зустрічаються в колекції документів.

Рядки відповідають документам в колекції, а стовпці відповідають термінам. До матриці застосовується сингулярне розкладання.

Сингулярне розкладання - це математична операція, розкладають матрицю на 3 складових. Сингулярне розкладання можна уявити у вигляді формули:

$$A = U \times S \times VT$$

де A - вихідна матриця, U і VT - ортогональні матриці, а S - діагональна матриця, значення, на діагоналі якої називаються сингулярними коефіцієнтами матриці A. Сингулярне розкладання дозволяє виділити ключові складові вихідної матриці.

Основна ідея ЛСА полягає в тому, що якщо в якості матриці A використовувалася терм-документа матриця, то матриця A *, що містить тільки k перших лінійно незалежних компонент, відображає основну структуру різних залежностей, присутніх у вихідній матриці. структура залежностей визначається ваговими функціями термів.

Як правило, вибір k залежить від поставленого завдання і підбирається емпірично. Якщо вибране значення k занадто велике, то метод втрачає свою потужність і наближається за характеристиками до стандартних векторних методів. Занадто мале значення k не дозволяє вловлювати відмінності між схожими термами або документами. Якщо ж необхідно вибирати значення k автоматично, то можна, наприклад, встановити граничне значення сингулярних коефіцієнтів і відкидати всі рядки і стовпці, відповідні сингулярним коефіцієнтами, що не перевищує дане порогове значення.

Схожість між будь-якою комбінацією термів і/або документів найчастіше обчислюють за допомогою скалярного твори їх векторів, однак на практиці кращий результат дає обчислення схожості з допомогою коефіцієнта кореляції Пірсона. ЛСА відображає документи і окремі слова в так зване «семантичний простір», в якому і проводяться всі подальші порівняння.

При цьому робляться такі припущення:

1. Документ це просто набір слів. Порядок слів у документах ігнорується. Важливо тільки те, скільки разів той чи інший слово зустрічається в документі.
2. Семантичне значення документа визначається набором слів, які, як правило, йдуть разом.
3. Кожне слово має єдине значення. Це, безумовно, сильне спрощення, але саме воно дозволяє вирішити проблеми.

TextRank і LexRank

TextRank і LexRank - це алгоритм ранжування на основі графіків. Він працює на основі PageRank алгоритму. Ребра представляють семантичну подібність, а вершини - речення.

3. Переваги та недоліки абстрактних та екстрактних методів

У таблиці 1 описані переваги та недоліки абстрактних та екстрактних методів автоматичного реферування тексту.

Табл. 1. Переваги та недоліки абстрактних та екстрактних методів

	Метод	**Переваги**	**Недоліки**
1	Метод оснований на теорії графів	Алгоритм підрахунку показників добре працює в аглютинативних мовах	Менше семантики
2	Класичний метод	Простота методу	Надмірне скорочення текстового документа
3	Tf-idf метод	Запити на основі речень	Надмірне скорочення текстового документа
4	Векторна модель	Кожне речення ранжується та визначається наскільки вони подібні до запиту. Відбираються підсумкові речення з високими коефіцієнтами	Це вимагає багато часу для обробки. Система для обробки семантичного змісту може використовуватися спеціальні теги. Вона не справляється із синонімією і полісемією
5	Методи машинного навчання	Надмірне скорочення може бути скореговане	Обчислювально складний метод
6	Латентно-семантичний метод	Відбувається екстракція на основі семантично пов'язаних речень, навіть якщо вони не мають спільних слів	Текст неоднорідний, розрахунок складний
7	Метод кластерів	Кластеризація може бути використана для групування подібних речень на різні теми та створення реферату	Складні алгоритми
8	Text Rank and LexRank	Базується на основі теорії графів	Менше семантики

Висновки

Постійне збільшення обсягу інформації призвело до необхідності автоматичного реферування тексту для полегшення завдання надання необхідної інформації користувачам. У даній статті були проаналізовані різні методи реферування текстових документів, такі як статистичний, семантичний методи, машинне навчання, та інші. Порівнюючи різні методи реферування, ми прийшли до такого висновку, що використовуючи різні методи гібридним шляхом, якість резюме буде більш ефективною, оскільки об'єднання, наприклад, двох методів призводить до усунення недоліків одного методу і використання переваг іншого. Також поєднання різних особливостей документа часто дає кращі результати при призначенні ваг реченням.

Список літератури

1. E. Hovy and C-Y Lin, (1997), “Automated Text Summarization in SUMMARIST”, Proceedings of the Workshop of Intelligent Scalable Text Summarization.
2. Vishal Gupta and Gurpreet Singh Lehal, (2010)“A Survey of Text Summarization Extractive Techniques”, Journal of Emerging Technologies in Web Intelligence, Vol. 2, No. 3.
3. Wong Kam-Fai. Extractive summarization using supervised and semi-supervised learning // Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008). Manchester, 2008. P. 985–992.
4. Yang, Ch. C Fractal Summarization for Mobile Devices to Access Large Documents on the Web [Текст]/ Ch. C. Yang, F.L. Wang//In Proc. of the WWW2003, Budapest, Hungary, 2003.–С. 134−139.
5. Edmundson, H.P. (1969), New Methods in Automatic Extracting, Journal of the ACM, 16(2):264-285.
6. Сорока М.Б. Національна система реферування української наукової літератури / НАН України, Нац. б-ка України імені В.І. Вернадського. – К.: НБУВ, 2002. – 209 с.
7. TF-IDF [Elektronnij resurs]. — Rezhym dostupu : https://uk.wikipedia.org/wiki/TF-IDF.
8. Kaikhah, K. (2004). Text Summarization Using Neural Networks. In proceeding of second conference on intelligent system, IEEE, vol. 1, pp. 40-44.

УДК 004.931

СТЕК Ю.А.
ХАЛУС О. А.

МЕТОД ВИЯВЛЕННЯ ЛЮДЕЙ ТА ОДНОЧАСНЕ ОЦІНЮВАННЯ ЇХ ПОЗ

В даній статті оглянуто проблему виявлення людей і одночасного оцінювання їх артикульованих поз. Оцінка людської пози визначається як локалізація анатомічних ключових точок або орієнтирів і вирішується з використанням різних методів, залежно від кінцевих цілей та зроблених припущень. Ціллю являється створення моделі, що може виявити людську позу в режимі реального часу з потокового відео.

КЛЮЧОВІ СЛОВА: система розпізнавання, виявлення людей, машинне навчання, YOLO.

In this article we examine the problem of identifying people and simultaneously assessing their articulated poses. The assessment of human posture is defined as the localization of anatomical key points or guidelines, and is solved using different methods, depending on the ultimate goals and assumptions. The goal is to create a model that can detect the human pose in real time from streaming video.

KEYWORDS: recognition system, detection of people, machine learning, YOLO.

1. Вступ

Проблема виявлення людей і одночасної оцінки їх артикульованих поз, стала дуже важливим практичним завданням в computer vision завдяки останнім досягненням у глибокому навчанні. Даний тип задач має широке застосування в таких областях, як аналіз спорту і взаємодія людини з комп'ютером, проте його вартість обчислення все ще може бути слабким місцем в системах реального часу. Оцінка людської пози визначається як локалізація анатомічних ключових точок або орієнтирів (або частин, кінцівок) і вирішується з використанням різних методів, залежно від кінцевої цілі та зроблених припущень:

- використання поодиноких або послідовних зображень в якості вхідних даних;
- використання (або відсутність) інформації про глибину зображення (RGB-D) в якості вхідних даних;
- локалізація деталей в 2D або 3D просторі;
- Оцінка одно- або багатоособових поз.

В даній статті, описано метод, що фокусується на оцінці 2D позиції з двома людьми з 2D нерухомого зображення.

Попередні підходи можна розділити на два типи: один виявляє людей, а потім виявляє позу однієї особи відповідно до кожної виявленної людини, а інший спочатку виявляє кінцівки, а потім розбирає їх відповідно для кожної людини. Це називаються підходами «зверху-вниз» і «знизу-вгору» відповідно. Дані підходи показують конкурентні результати як і в продуктивності, так і в точності визначення. Однак час виконання підходу «зверху-вниз» пропорційний кількості людей, що робить виконання в реальному часі викликом, тоді як підходи «знизу-вгору» працюють з кінцівками і виконують розбір кінцівок окремих людей. Крім того, більшість сучасних технологій призначені для прогнозування піксельної карти частин зображення. Ці карти зумовлюють використання конвуляційної нейронної мережі (CNN) для вилучення карт з більш високою роздільною здатністю, які необхідні для підтримки надійності та прискорення архітектур (наприклад, скорочення архітектури).

2. Метод

Ми використовуємо переваги YOLO [1, 2], одного з фреймворків ПЗ (передачення зони; тобто деякого регіону на картинці, де може бути кінцівка), і застосовуємо його концепцію до завдання визначення людини. Модель побудована з однієї CNN і створює колекцію фіксованих розмірів ПЗ для кожної цілі виявлення (особи або кожної кінцівки) над вхідним зображенням. CNN ділить вхідне зображення на сітку розмірності $H \times W$, кожна комірка якої відповідає блоку зображення, і

створює набір ПЗ-виявлення $\{B^i_k\}_{k\in\mathcal{K}}$ для кожної комірки сітки $i \in \mathcal{G} = \{1,\ldots, H \times W\}$. Тут $\mathcal{K} = \{0, 1,\ldots, K\}$ є набір індексів цілей виявлення, а K - кількість частин. Індекс класу, що представляє загальні екземпляри людини, задається k = 0 в $\mathcal{K}$.

$\{B^i_k\}_{k\in K}$ відображає дві ймовірності, беручи до уваги вірогідність обмежувального блоку (границі виявленої кінцівки) і координати, ширину і висоту обмежувального блоку.

$$B^i_k = \{p(R \mid k,i), p(I \mid R,k,i), o^i_{x,k}, o^i_{y,k}, w^i_k, h^i_k\}, \quad (1)$$

де R і I – бінарні випадкові величини. Тут $p(R \mid k,i)$ є ймовірністю, яка представляє клітинку i сітки «відповідальну» за виявлення k. Якщо центр зображення обмежувального блоку k потрапляє в комірку сітки, ця клітинка сітки є "відповідальною" за виявлення k. $p(I \mid R,k,i)$ - умовна ймовірність, яка представляє, наскільки добре обмежувальний блок передбачений в i відповідає k і контролюється перетином над об'єднанням (IoU) між передбаченим обмежувальним блоком і дійсним обмежувальним блоком.

Координати $o^i_{x,k}, o^i_{y,k}$ представляють центр обмежувального блоку відносно межі сітки із шкалою, нормалізованої по довжині клітинок. w^i_k та h^i_k нормалізовані відповідно до ширини та висоти зображення. Обмежувальні блоки навколо людини можуть бути представлені як прямокутники навколо всього тіла або голови. В даному методі, деталі виявляються сіткою і розміри блоків контролюються пропорційно масштабам людини, наприклад, одна п'ята довжину верхньої частини тіла або половина довжини голови.

І навпаки, для кожної клітини сітки i, розташованої в точці x, CNN також створює набір виявлених кінцівок, $\{C_{k_1,k_2}\}_{(k_1,k_2)} \in L$, де L - сукупність пар індексів відображаючих виявленні цілі, тобто кінцівки. C_{k_1,k_2} - набір ймовірностей, що представляє наявність кінцівки в клітинці і задається наступним чином:

$$C_{k_1,k_2} = \{p(C \mid k_1, k_2, x, x + \Delta x)\}_{\Delta x \in X}, \quad (2)$$

де C - бінарна випадкова величина. $p(C \mid k_1, k_2, x, x + \Delta x)$ - наявність кінцівки, представлена у вигляді спрямованого з'єднання від обмежуючого блоку k_1, передбаченого в x до того k_2, передбаченого в $x + \Delta x$, як показано на рис. 3.

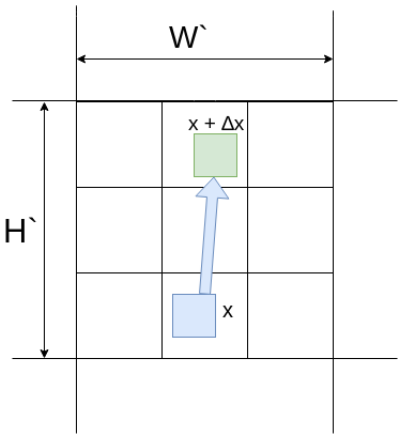


Рис. 3. Виявлення кінцівок. Синя стрілка відображає кінцівку (спрямоване з'єднання) виявлення якої відбувається за формулою $\boldsymbol{p(C \mid k_1, k_2, x, x + \Delta x)}$.

Тут припускаємо, що всі кінцівки від x досягають тільки локальних H' × W', і $\mathcal{X}$ визначає як набір кінцевих переміщення з x, що задається формулою 3.

$$X = \{\Delta x = (\Delta x, \Delta y) \mid |\Delta x| \le W' \wedge |\Delta y| \le H'\}, \quad (3)$$

Тут Δx - положення відносно x і, отже, $p(C \mid k_1, k_2, x, x + \Delta x)$ може бути незалежно оцінено на кожній клітинці сітки з використанням CNN завдяки їх характеристиці інваріативності переведення.

Кожне з вищезазначених передбачень відповідає кожному каналу в глибину вихідного 3D-тензора, створеного CNN. Нарешті, CNN виводить тензор. Під час навчання ми оптимізуємо наступні, багаточастинні функція втрати:

$$\lambda_{resp.} \sum_{i \in G} \sum_{k \in K} \{\delta^i_k - \hat{p}(k,i)\}^2$$

$$+ \lambda_{IoU} \sum_{i \in G} \sum_{k \in K} \delta^i_k \{p(I \mid R,k,i) - \hat{p}(I \mid R,k,i)\}^2$$

$$+\lambda_{coor.}\sum_{i\in G}\sum_{k\in K}\delta_k^i\{\left(o_{x,k}^i-\hat{o}_{x,k}^i\right)^2+\left(o_{y,k}^i-\hat{o}_{y,k}^i\right)^2\}$$

$$+\lambda_{size}\sum_{i\in G}\sum_{k\in K}\delta_k^i\{\left(\sqrt{w_k^i}-\sqrt{\hat{w}_k^i}\right)^2+\left(\sqrt{h_k^i}-\sqrt{\hat{h}_k^i}\right)^2\}$$

$$+\lambda_{limb}\sum_{i\in G}\sum_{\Delta x\in X}\sum_{(k_1,k_2)\in L}\max\left(\delta_{k_1}^i,\delta_{k_2}^i\right)\left\{\delta_{k_1}^i,\delta_{k_2}^i-\hat{p}(C\mid k_1,k_2,x,x+\Delta x)\right\}^2, \quad (4)$$

де $\delta_k^i \in \{1, 0\}$ є змінною, яка вказує на те, що i відповідає за k всього єдиної людини, j - індекс клітинки сітки, розташованої на $\boldsymbol{x + \Delta x}$, і ($\lambda_{resp.}$, λ_{IoU}, $\lambda_{coor.}$, λ_{size}, λ_{limb}) є вагами для кожної функції втрати.

У роботі є дві важливі особливості, що дають можливість опрацьовувати задачі в режимі реального часу. По-перше, для отримання остовного дерева скелетону вибирається мінімальна кількість ребер, чиї вузли і ребра являють собою об'єднані ПЗ підмножини цілей виявлення (кінцівок) і виявленних кінцівок серед них, відповідно, замість використання повного графа. Це дерево складається з спрямованих ребер і його вершини належать до екземпляра людини. По-друге, проблема зіставлення далі розкладається на набір відповідних дводольних під-задач і відповідності в незалежно визначених суміжних вершинах дерева. Даний підхід істотно поліпшує оцінки методу: швидкодію, значення функцій втрат.

2.2. Довірчі оцінки

Враховуючи об'єднані ПЗ цілей виявлення, ми визначаємо довірчу оцінку для виявлення n-го ПЗ з k наступним чином:

$$D_k^n = p(R \mid k,n), p(I \mid R,k,n), \quad (5)$$

Кожна ймовірність у правій частині рівняння (5) визначаються через B_k^i у формулі (1), $n \in N = \{1,\ldots, N\}$, де N - кількість об'єднаних ПЗ кожної з виявлених цілей. Крім того, довірчі оцінки кінцівки, тобто спрямованих з'єднань від n_1-го ПЗ k_1, передбаченого в x до n_2-го ПЗ k_2, передбаченого в $x + \Delta x$, визначаються шляхом використання рівняння (2) наступним чином:

$$E_{k_1,k_2}^{n_1,n_2} = p(C \mid k_1,k_2,x,x+\Delta x), \quad (6)$$

2.3. Об'єднання частин

Об'єднання частин, визначається як оптимальна задача для безлічі всіх можливих об'єднань,

$$Z = \{Z_{k_1k_2}^{n_1n_2} \mid (k_1,k_2) \in L, n_1 \in N_1, n_2 \in N_2\}, \quad (7)$$

що максимізує довірчу оцінку, яка підвищує загальну ймовірність для всіх можливих виявлень кінцівок,

$$F = \prod_L \prod_{N_1} \prod_{N_2} \left(E_{k_1k_2}^{n_1n_2}\right)^{Z_{k_1k_2}^{n_1n_2}}, \quad (8)$$

Тут $Z_{k_1k_2}^{n_1n_2}$ є бінарною змінною, яка вказує, чи n_1-й ПЗ з k_1 і n_2-й ПЗ з k_2 пов'язані і задовольняють умові

$$\sum_{N_1} Z_{k_1k_2}^{n_1n_2} = 1 \wedge \sum_{N_2} Z_{k_1k_2}^{n_1n_2} = 1, \quad \forall n_1 \in N_1, \forall n_2 \in N_2, \quad (9)$$

Використання рівняння (9) гарантує, що не багато ребер мають спільну вершину, тобто, що ПЗ не з'єднаний з різними множинами ПЗ. У цьому графі узгодження задачі, вузлами графа є всі об'єднані ПЗ цілей виявлення, ребрами є всі можливі зв'язки між ПЗ, які складають кінцівки, і довірчі оцінки кінцівки дають ваги для ребер. Мета - знайти відповідність у дводольному графі, як підмножину ребер, вибраних з максимальною вагою.

Людина використовуються як коренева частина, а кожна виявлена частина тіла, як маршрут в графі. Підпроблеми відповідності визначаються для кожної пари цілей виявлення (k_1, k_2), так, щоб знайти оптимальне значення для набору зв'язків між k_1 і k_2,

$$Z_{k_1k_2} = \{Z_{k_1k_2}^{n_1n_2} \mid n_1 \in N_1, n_2 \in N_2\}, \quad (10)$$

де

$$\{Z_{k_1k_2}\}_{(k_1,k_2)\in L} = Z, \quad (11)$$

Отримуємо оптимальне значення $\hat{Z}_{k_1k_2}$ наступним чином:

$$\hat{Z}_{k_1k_2} = F_{k_1k_2} \quad (12)$$

де

$$F_{k_1k_2} = \prod_{N_1} \prod_{N_2} \left(S_{k_1k_2}^{n_1n_2}\right)^{Z_{k_1k_2}^{n_1n_2}} \quad (13)$$

Тут вершини k_1 ближчі до тих, що мають екземпляри людини на маршруті графу, ніж у k_2 і

$$S_{k_1k_2}^{n_1n_2} = \{D_{k_1}^{n_1} E_{k_2k_1}^{n_2n_1} D_{k_2}^{n_2}, \; if \; k_1 = 0 \; S_{k_0k_1}^{\hat{n}_0n_1} E_{k_2k_1}^{n_2n_1} D_{k_2}^{n_2}, \quad else \quad (14)$$

$k_0 \neq k_2$ вказує іншу ціль виявлення пов'язану з k_1. $\hat{n}_0$- індекс ПЗ k_0, який пов'язаний з n_1-й ПЗ з k_1 і задовольняє

$$Z_{k_0k_1}^{\hat{n}_0n_1} = 1, \quad (15)$$

Ця оптимізація з використанням рівняння (14) необхідні для обчислення частин (кінцівок), які визначені як ті, що належать даній людині. Нарешті, з усіма оптимальними значеннями ми можемо зібрати з'єднання які поділяють однакові ПЗ на повні тіла людей.

Результати:

Даний метод досягає найвищої продуктивності для верхніх частин тіла. Загальна тривалість обробки досягає до 5,6 мс (180 FPS).

Висновки

Описаний метод здатний виявити людей і одночасно висунути оцінки їх 2D поз, вирізняючи їх з 2D нерухомого зображення. Основні нововведення для підвищення швидкості / точності полягають у запровадженні найсучаснішої парадигми виявлення об'єктів з сценарієм виявлення позиції «знизу-вгору». На додаток, кінцівки виявляються безпосередньо за допомогою CNN. Експериментальні результати підтверджують, що даний метод має порівняну точність з найсучаснішими підходами «знизу-вгору» і є швидшим. Можливо покращити продуктивність для просторових обмеження, спричинених грубими прогнозами сітки.

Список літератури

1. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR. (2016)
2. Redmon, J., Farhadi, A.: YOLO9000: Better, faster, stronger. In: CVPR. (2017)

УДК 681.3.01

ШЕХЕТ Г.О.

СИСТЕМА АНАЛІЗУ КОНТЕНТУ ПОТОКОВОГО ВІДЕО ТА ФОРМУВАННЯ РЕЛЕВАНТНОЇ КОНТЕКСТНОЇ РЕКЛАМИ

У статі представлено алгоритм вирішення задачі аналізу контенту відеопотоку у реальному часі, для пошуку релевантної контекстної реклами. Даний алгоритм здійснює аналіз орієнтуючись на три складові відеопотоку: метадані, кадри з відео та його аудіо доріжку. На основі розробленого алгоритму побудований прототип інформаційної системи, що базуючись на контенті відеопотоку, пропонує користувачеві релевантну контекстну рекламу.

АНАЛІЗ КОНТЕНТУ ПОТОКОВОГО ВІДЕО, КЛАСИФІКАЦІЇ ТА РОЗПІЗНАВАННЯ ОБРАЗІВ У ВІДЕО, РОЗПІЗНАВАННЯ ОБРАЗІВ ЗА ДОПОМОГОЮ КОМП'ЮТЕРНОГО ЗОРУ, АНАЛІЗ ЗВУКОВОГО ПОТОКУ ВІДЕО

In this paper, we present an algorithm for solving the problem of analyzing the content of video stream in real time, for searching for relevant contextual advertising. This algorithm carries out an analysis focusing on three components of the video stream: metadata, footage from the video and its audio track. The prototype of an information system was developed based on the suggested algorithm which based on the content of a video stream is suggesting the user the relevant contextual advertising.

ANALYSIS OF THE STREAMING VIDEO CONTENT, CLASSIFICATIONS AND RECOGNITION OF VARIABLES IN VIDEO, RECOGNITION OF PICTURES BY COMPUTER VISION, ANALYSIS OF VIDEO SOUND FLOW

1. Вступ

В останні роки одним з найпопулярніших джерел інформації стає відео. Існує безліч різноманітних онлайн кінотеатрів та відео сервісів, які щодня надають можливість мільйонам користувачів дивитись фільми, телепередачі та шоу на будь-який смак. Основна частина доходів даних систем йде саме з реклами. Зазвичай, глядачам під час перегляду пропонують різноманітні рекламні оголошення, які, на жаль, не відповідають змісту відео. Через це реклама виглядає набридливою і не є цікавою глядачу, а отже не приносить очікуваного доходу рекламодавцям. Таким чином вирішення задача пошуку і підбору рекламних оголошень, які будуть відповідати змісту відео, є вартою уваги і дуже актуальною у наш час.

У статі представлено алгоритм вирішення задачі аналізу контенту відеопотоку у реальному часі, для пошуку релевантної контекстної реклами. Даний алгоритм здійснює аналіз орієнтуючись на три складові відеопотоку: метадані, кадри з відео та його аудіо доріжку.

На основі розробленого алгоритму побудований прототип інформаційної системи, що базуючись на контенті відеопотоку, пропонує користувачеві релевантну контекстну рекламу.

2. Аналіз сучасних методів та досліджень в області аналізу контенту відеопотоку.

Існує багато методів та досліджень, які намагаються вирішити задачу аналізу контенту відеопотоку. Серед, них є ряд алгоритмів, які аналізують відео за допомогою семантичного аналізу [1-2], шляхом ускладнення формату кодування чи дослідженням його метаданих [3], здійснюють пошук головних об'єктів відео шляхом відстеження ключових кадрів [4] чи за допомогою комп'ютерного зору [5], намагаються знайти контент шляхом аналізу аудіо доріжки відео [6].

Проте, більшість із запропонованих підходів можуть працювати повільно у реальному часі або використовуються тільки для розв’язання окремих, специфічних задач.

3. Алгоритм розв’язання задачі

На основі досліджених підходів був запропонований власний алгоритм, який вирішує задачу аналізу контенту відеопотоку у реальному часі і аналізує контент відео, орієнтуючись на три фактори: метадані, кадри з відео та його аудіо доріжку.

Розроблений алгоритм розбиває відеопотік на частини, після яких буде показана реклама, і паралельно аналізує контент для кожної з них у чотири етапи.

На першому етапі, відбувається розкадровка відео та виконується аналіз отриманих кадрів, шляхом пошуку головних об’єктів з використанням алгоритмів аналізу зображень та комп’ютерного зору.

Другий етап, складається з аналізу і розпізнавання мови з аудіо доріжки відео і пошуку ключових слів в отриманому тексті.

На третьому етапі, алгоритм аналізує метадані відео і знаходить ключові слова у них.

На останньому етапі відбувається пошук максимально наближеної реклами до контенту відеопотоку на основі цільовій функції, яка базується на даних про об’єкти отримані підчас попередніх розрахунків.

4. Алгоритм пошуку та розпізнавання головних об’єктів у кадрах з відео

Підходи пошук, та розпізнавання головних об’єктів у відео найчастіше складаються з двох етапів. На першому етапі виконується фільтрація зображення, а на другому його безпосередній аналіз.

Проте існують методи комп’ютерного зору які значно полегшують виконання обох функції. Але найчастіше дані операції можуть виконуватись дуже повільно.

Саме тому ключова ідея розробленого алгоритму полягає в оптимізації процесу аналізу методами комп'ютерного зору, шляхом знаходження спільних предметів і аналізу тільки знайдених фрагментів з кадрів.

Дана оптимізація, відбувається за рахунок операції пошуку контурів об'єктів у кадрах за допомогою оператора Превітта. Потім, на основі знайдених контурів об'єктів, виконується операція пошуку спільних предметів у кадрах за допомогою перцептивних хешів та дискретного косинусоїдального перетворення (ДКП) – методу pHash. В результаті, якого отримані хеші порівнюються за допомогою відстані Хемінга. Врешті решт на тільки виконується безпосереднє розпізнавання схожих об'єктів за допомогою комп'ютерного зору. Оскільки тепер ми виконуємо дану операцію не до всіх кадрів, а тільки до ряду знайдених об'єктів, які повторюються у кадрах. Розглянемо більш детально даний алгоритм.

4.1. Метод розпізнавання контурів об'єктів за допомогою оператора Превітта

Для оптимізації розпізнавання об'єктів у кадрах з відео слід знайти фрагменти об'єктів у зображеннях, за допомогою знаходження їх контурів.

Зазвичай, задача пошуку кордонів зводиться до задачі перепадів у зображеннях – множини пікселів, яка лежить між двох областей. Пошук перепадів визначається на основі в становлення порогу для певної точки зображення. Дана точка визначається, як певна точка перепаду за умови, якщо її двовимірна похідна першого порядку перевищує певний поріг. В результаті множина таких точок перепаду яскравості і називається контуром.

Отож, для вирішення поставленої задачі, знайдемо першу похідна функції $f(x)$, як різницю значень сусідніх елементів:

$$\frac{\partial y}{\partial x} = f(x+1) - f(x)$$

Аналогічно, друга похідна визначається як різниця сусідніх значень першої похідної:

$$\boldsymbol{\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)}$$

Обчислення першої похідної цифрового зображення засноване на різних дискретних наближень двовимірного градієнта. За визначенням, градієнт зображення $f(x,y)$ в точці (x,y) - це вектор:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Важливу роль при виявленні контурів грає величина модуля вектору, яка визначається за формулою:

$$\nabla f = |\nabla f| = \sqrt[2]{G_x^2 + G_y^2}$$

А також напрямок цього вектору. Він позначається $\alpha(x,y)$ і є кутом між напрямком вектора ∇f в точці (x,y) і віссю x і дорівнює:

$$\alpha(x,y) = arctg\left(\frac{G_y}{G_x}\right)$$

Звідси можна знайти напрям контуру в точці (x,y) , яке перпендикулярно напрямку вектора градієнта в цій точці.

Обчислення градієнта зображення полягає в отриманні величин приватних похідних для кожної точки. Одним з способів знаходження перших похідних в точці є метод використання масок або процес просторовою фільтрації. Дані маски представляються у вигляді квадратний матриць довільної довжини.

Процес просторової фільтрації, у зображеннях, здійснюється шляхом переміщення маски фільтра від точки до точки зображення.

Тепер безпосередньо розрахуймо контури для нашого зображення з використанням так званої маски. Нехай маємо матрицю 3 на 3, яка складається з значень яскравості в області деякого елемента зображення:

$$\begin{pmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{pmatrix}$$

Один з способів знаходження перших приватних похідних в точці z_5 полягає в застосуванні наступного перехресного градієнтного оператора Перевіта:

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

та

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

У цих формулах різниця між сумами по верхньому і нижньому рядках в заданій області є наближеним значенням похідної по осі x, а різниця між сумами по першому і останньому стовпцях цієї області – похідною по осі y. Для реалізації цих формул

використовується оператор, описуваний оператором Превітта:

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

та

$$G_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Використавши оператор Превітта у формулі градієнту ми отримаємо зображення з кадрів в яких виділені тільки контури об'єктів. За допомогою яких можна виділити області у кадрах в яких знаходяться шукані об’єкти.

4.2. Алгоритм пошуку схожих предметів на основі перцептивних хешів та ДКП

Щоб знайти схожі об’єкти по знайденим контуром можна використати частотний аналіз зображень, а саме використати перцептивні хеш-алгоритми для знаходження схожих об’єктів. Для їх знаходження у зображені існує ряд поширених алгоритмів.

Перцептивні хеш-алгоритми описують клас функцій для генерації порівнянних хешів, які можна отримати з фрагментів зображень, які в результаті можна порівнювати один з одним. На відміну від криптографічних хешів, перцептивні хешів не обов’язково повинні бути однаковими, щоб сказати що данні збігаються. Тому їх порівнюють між собою за ступенем відмінностей наборів даних.

Оскільки у зображеннях високі частоти забезпечують деталізацію, а низькі частоти показують структуру. У великій деталізованій фотографія міститься багато високих частот, натомість у дуже маленькій немає деталей, а отже вона цілком складається з низьких частот, які дуже просто відтворити у вигляді хеш-функції.

Одним з найкращих методом хешування можна вважати pHash, який виявляє стійкість до малих поворотів, розмиття і сходження зображення, а також є досить швидким. В основі даного методу використовується дискретне косинусоїдальне перетворення (ДКП) – одне з ортогональних перетворень, тісно пов'язане з дискретним перетворенням Фур'є (ДПФ). ДКП, як і будь-яке перетворення Фур'є, представляє функцію у вигляді суми синусоїд з різними частотами і амплітудами. ДКП використовує тільки косинусні функції, на відміну від ДПФ, яке використовує і косинусні, і синусні функції.

Для вирішення поставленої задачі використаємо другий тип ДКП. Нехай $x[m]$, де $m = 0, \dots, N-1$ – послідовність сигналу довжин N. Визначимо другий тип ДКП, як

$$X[n] = \sqrt[2]{\frac{2}{N}} \sum_{m=0}^{N-1} x[m] \cos\left(\frac{(2m+1)\, n\, \pi}{2N}\right)$$

де $n = 0, \dots, N-1$. Цей вислів можна переписати як:

$$X[n] = \sum_{m=0}^{N-1} c[n,m]\ x[m]$$

де $c[n,m]$ - елемент матриці ДКП на перетині рядка з номером n і стовпці з номером m. ДКП матриця визначається як:

$$X[n] = \sqrt[2]{\frac{2}{N}} \sum_{m=0}^{N-1} \cos\left(\frac{(2m+1)\, n\, \pi}{2N}\right)$$

Дана матриця дуже зручна, для обчислення ДКП. ДКП може бути обчислена заздалегідь, для будь-якої необхідної довжини.

Таким чином ДКП може бути представлена у вигляді:

ДКП = M × I × M'

де M – ДКП матриця, I - зображення квадратного розміру, M'- зворотна матриця.

Тепер побудуємо хеш з зображення за допомогою ДКП, методу pHash:

КРОК 1. Зменшити розмір зображення до розміру в діапазоні 32x32.

КРОК 2. Виконаємо операцію знебарвлення зображення.

КРОК 3. Визначити середнє значення кольору матриці.

КРОК 4. Запустити ДКП для отриманої матриці.

КРОК 5. Будуємо хеш для отриманої матриці.

Далі для порівняння отриманих хешів використаємо функцію відстані Хеммнга. У даному випадку відстанню Хеммінга де $d_H[X_i, X_j]$ для двох хешів фрагментів зображень X_i та X_j довжини p, можна вважати число позицій, в яких вони відмінні:

$$d_H[X_i, X_j] = \sum_{s=1}^{p} sign \left| x_i^{(s)} - x_j^{(s)} \right|$$

Після знаходження спільні об’єктів у кадрах, слід використати методи комп’ютерного зору для їх безпосереднього розпізнавання.

5. Алгоритм аналізу аудіо доріжки відео

Етап аналізу аудіо доріжки відео складається з двох частин. На першому кроці відбувається розпізнавання мови з аудіо доріжки відео за допомогою алгоритму який використовує словник на основі MFCC. А на другому кроці, виконується пошуку ключових слів в отриманому тексті на основі алгоритму TF-IDF. Розглянемо більш детально дані кроки.

5.1. Алгоритм розпізнавання мови з аудіо доріжки на основі MFCC

При розпізнаванні мови в першу чергу необхідно розбити її на слова. Існує багато способів її розпізнавання. У даній роботі використаємо алгоритм, який використовує словник на основі MFCC (мел кепстральних коефіцієнтів).

MFCC – вектор з тринадцяти дійсних чисел. Він є енергією спектру сигналу. Даний метод враховує хвильову природу сигналу, мел-шкала виділяє найбільш суттєві частоти, які сприймаються людиною, а кількість MFCC коефіцієнтів можна задати будь-яким числом, що дозволяє стиснути фрейм і зменшити кількість оброблюваної інформації.

За базу навчання будемо використовувати безліч файлів, кожен з яких представляє собою набір MFCC-векторів, отриманих з фонограми із записом того чи іншого слова. При цьому файли із записом одного і того ж слова повинні бути об'єднані в одну групу.

Алгоритм складається з наступний етапів:

КРОК 1. Знаходимо супер вектор середніх для всієї бази навчання за допомогою алгоритму K-середніх.

КРОК 2. Для кожного файлу бази знаходимо власні середні значення за формулою:

$$M_k = a * M_{k0} + (1 - a) * M'_k, k = 1 \dots K$$

де M_{k0} – середнє значення, знайдене підчас першого кроку, а M'_k – середнє значення, отримане в результаті застосування однієї ітерації алгоритму K-середніх для MFCC-векторів файлу з використанням в якості початкового значення M'_{k0} ,

$$a = \frac{R}{(R + N_k)},$$

де R - коефіцієнт «чутливості», N_k - число MFCC-векторів, що відповідають середнім значенням M'_k. Знайдені таким чином середні значення будемо називати адапрітованими середніми значеннями.

КРОК 3. Маючи тепер замість вихідних фонограм адаптовані супервектора середніх, проводимо LDA для N класів (кожен клас відповідає одному слову).

В результаті ми повинні отримати матрицю, що складається з векторів нового базису, при проекції на який вихідні адаптовані супер вектора середніх повинні достатньо добре розділятися.

КРОК 4. Проектуємо всі адаптовані супер вектора середніх на новий базис і знаходимо середні значення і СКО (середнє квадратичне відхилення) проекцій для кожного класу.

КРОК 5. Для визначення приналежності тестової фонограми того чи іншого класу (тобто розпізнавання), виконуємо для неї кроки 2 і 4, далі знаходимо відстані отриманої проекції до середніх значень всіх класів (можна додатково унормувати їх на відповідне СКО). Мінімальна відстань і буде відповідати класу, до якого належить тестова фонограма.

5.1. Алгоритм пошуку ключових слів у тексті з звукової доріжки на основі TF-IDF

Для знаходження ключових слів у розпізнаному тексті з звукової доріжки використаємо алгоритм на основі TF-IDF, який використовується для отримання ключових слів (термів). Взагалі TF-IDF – статистичний показник, який використовується для оцінки значимості слів у тексті, який є частиною колекції текстів. Значимість слова прямо-пропорційна кількості вживань цього слова в тексті, та обернено-пропорційна кількості вживань цього слова в інших текстах колекції. Таким чином:

$$TF(d, t) = \frac{n_{d,t}}{n_d}$$

$$IDF(t) = \frac{N}{\left|d_j \supset t\right|}$$

Де $n_{d,t}$ – кількість появ слова t в тексті d, n_d – кількість слів в тексті d, $\left|d_j \supset t\right|$ – кількість текстів в яких зустрічається слово t.

Для отримання значимості – оцінки цього слова будемо використовувати модифікацію функції ранжирування Okapi BM25, яка використовується пошуковими системами, щоб оцінювати документи на відповідність пошуковому запиту:

$$S = IDF(t)\frac{TF(d,t)\ (k_1+1)}{TF(d,t)+k_1\left(1-b+b\ \frac{|d|}{avgl}\right)}$$

де $avgl$ – середня довжина текстів у колекції, а k_1 та b – вільні коефіцієнти.

Перед застосуванням TF-IDF алгоритму, текст попередньо обробляється: видаляються «стоп-слова», а також слова що залишилися приводяться до спільної основи(шляхом відкидання суфікса та(чи) основи) за допомогою стемінга. Кількість ключових слів, які відбираються в результаті – це 1/10 від загальної кількості слів, тому що щільність слів у отриманому тексі з звукової доріжки відео дуже висока.

Алгоритм TF-IDF складається з наступних кроків:

Крок 1. Для кожного слова в тексті порахувати частоту появи цього слова.

Крок 2. Для кожного слова визначити його значимість на основі функції Okapi BM25.

Крок 3. Упорядкувати слова за спаданням їх значимості.

Крок 5. Повернути 1/10 третину фраз з упорядкованого списку.

6. Математична постановка задачі

На останньому етапі виконується пошук відповідного рекламного оголошення з множини контекстних рекламних оголошень, для кожного з яких існує множина предметів, які рекламує дане оголошення. На вхід системи подається відео потік, а також предмети які були знайдені у кадрах з відео, ключові слова з аудіо доріжки відео та його мета теги. В результаті аналізу ми проверстаємо набір контекстних оголошень, які будуть пропонуватися глядачу кожні t одиниць часу.

Тепер, сформулюємо поставлену задачу для довільного проміжку часу t, за який глядачу слід показати контекстну рекламу.

Дано:

$\{M_i\}$ – множина предметів, яка відповідає певній контекстній рекламі, де i – індекс контекстного рекламного оголошення, а $M_i = \{m_{i1}, m_{i2}, \dots\}$;

$\{m_{ij}\}$ – назва j-ий предмета у множині предметів рекламного оголошення M_i, де j – індекс предмета у множині предметів в контекстному рекламному оголошенні M_i;

$\{S\}$ – множина спільних предметів у кадрах за проміжок часу t у відео потоці (де $S = \{s_1, s_2, \dots\}$);

$\{s_k\}$ – назва k-ий предмет з множині спільних предметів у кадрах S, де k – індекс предмета у множині спільних предметів у кадрах S;

$\{n_k\}$ – кількість k-тих предметів у множині спільних предметів у кадрах S;

$\{L\}$ – множина предметів у звуковому контенті за проміжок часу t у відео потоці, де $L = \{l_1, l_2, \dots\}$;

$\{l_f\}$ – назва f-того предмету у множині предметів у звуковому контенті L, де f – індекс предмета у множині звукового контенту за проміжок часу L;

$\{b_f\}$ – кількість f-тих предметів у множині предметів у звуковому контенті L;

$\{R\}$ – множина предметів з мета даних у відео, де $R = \{r_1, r_2, \dots\}$;

$\{r_e\}$ – назва e-того предмету у множині мета даних у відео R, де e – індекс предмета у множині множині мета тегів відео R;

Змінні: $\{x_{ij}\}$ – відповідність j-того предмета з множини предметів i-того рекламного оголошення M_i, до множини спільних предметів у кадрах за проміжок часу t у відео потоці S:

$$x_{ij} = \begin{cases} 0, \text{ якщо } m_{ij} \notin S; \\ a_k, \text{ якщо } \exists\, k, \text{ що } s_k = m_{ij},\ s_k \in S, m_{ij} \in S; \end{cases}$$

$\{y_{ij}\}$ – відповідність j-того предмета з множини предметів i-того рекламного оголошення M_i, до множини предметів у звуковому контенті за проміжок часу t у відео потоці L:

$$y_{ij} = \begin{cases} 0, \text{ якщо } m_{ij} \notin L; \\ b_f, \text{ якщо } \exists\, f, \text{ що } l_f = m_{ij},\ l_f \in L, m_{ij} \in L; \end{cases}$$

$\{z_{ij}\}$ – відповідність j-того предмета з множини предметів i-того рекламного оголошення M_i, з мета даних у відео R:

$$z_{ij} = \begin{cases} 0, \text{ якщо } m_{ij} \notin R; \\ 1, \text{ якщо } \exists\, e, \text{що } r_e = m_{ij},\ r_e \in R, m_{ij} \in R; \end{cases}$$

Цільова функція:

Максимізувати міру (відсоток) попадання предметів множина предметів, яка відповідає певній контекстній рекламі до множини предметів з відео кадрів, звукового потоку та мета даних відео:

$$\sum_{j=1}^{\infty}\left(x_{ij} + y_{ij} + z_{ij}\right) \rightarrow max$$

Обмеження:

$0 < i, j, k, f, e, t < \infty$;

$1 < a_k, b_f < \infty$;

$i, j, k, f, e, t, a_k, b_f \in N$;

$i, j, k, f, e, a_k, b_f \in Z$;

7. Система аналізу контенту потокового відео та пошуку релевантної контекстної реклами

Підчас виконання даної роботи на основі розробленого алгоритму був створений прототип системи, яка підчас програвання відео, розпізнає його контент і пропонує глядачу релевантну контекстну рекламу до нього.

Дана інформаційна система, представлена у вигляді клієнт-серверної архітектури (Рис. 1). Розроблений сервер, може спілкуватися з клієнтами (застосуваннями чи браузерами) шляхом сокетів. Підчас підключення каналу, клієнт відправляє серверу посилання на відео потік, і сервер автоматично починає його обробляти. Після знаходження реклами для певного проміжку часу, сервер відправляє на клієнт по відкритому каналу сокета посилання на релевантну контекстну рекламу і час коли слід її показати користувачу.

В якості мов написання системи обрані Node.JS та C++. Так серверне застосування написане на NodeJS, а різноманітні розрахунки робляться на C++, для пришвидшення обчислень.

Для роботи з відео потоком була використанням бібліотеки FFmpeg, яка дозволяє виконувати розкадровку відео, робити різноманітні маніпуляції з кадрами, а також діставати аудіо доріжку з відеоряду.

Для розпізнавання і класифікації предметів була обрана бібліотеки OpenCV, яка використовує методи комп'ютерного зору підчас виконання даних операцій.

В якості бази даних рекламних оголошень використовується не реляційна БД – MongoDB.

На останок, для пришвидшення виконання розрахунків був обраний сервіс AWS Lambda, на якому і виконуються обчислення програми.

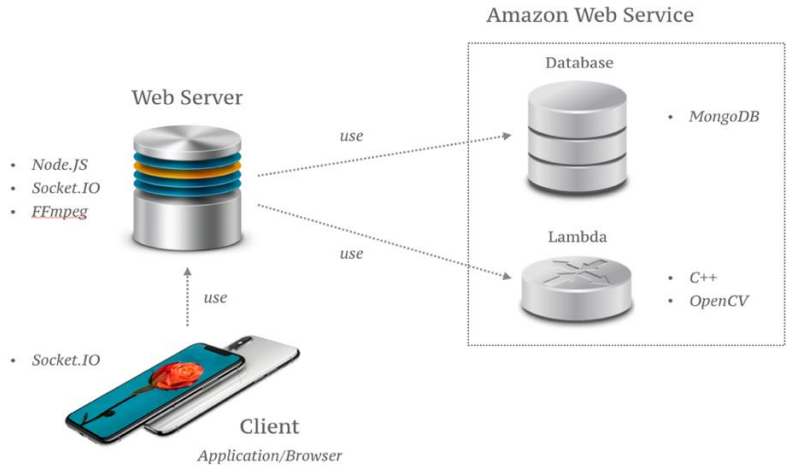


Рис.2. Схема архітектури системи аналізу контенту потокового відео та пошуку релевантної контекстної реклами

Висновок

Проведено фундаментальне дослідження в області відео аналізу. Розроблений ефективний алгоритм для швидкого пошуку ключових об'єктів у відео, який базується на метаданих, кадрах з відео та його аудіо доріжці. Розроблений прототип інформаційної системи, що базуючись на контенті відеопотоку, пропонує користувачеві релевантну контекстну рекламу.

Список використаних джерел

1. Чонг-ва Нгою. Recent advances in content-based video analysis. International Journal of Image and Graphics. Hong Kong University of Science & Technology. 2011. № 3, С. 445-468.
2. Ламберто Баллан. Event detection and recognition for semantic annotation of video. Multimedia Tools and Applications. 2011. № 51, С. 279–302.
3. Лян Чжао, Чжихай Він. Real-Time Moving Object Segmentation and Classification From HEVC Compressed Surveillance Video. IEEE Transactions on Circuits and Systems for Video Technology. Червень 2016. № 6, С: 1346 - 1357.
4. Йонг Джає Лі. Key-segments for video object segmentation. ICCV '11 Proceedings of the 2011 International Conference on Computer Vision. Університет Тихасу Остін, листопад 2011. С: 1995-2002.
5. Стюарт Джексон. Flexible, Mobile Video Camera System and Open Source Video Analysis Software for Road Safety and Behavioral Analysis. Університет Макгілла, Канада. Січень 2013. № 1. С: 90-98.
6. Вей Цзян, Олександр Луї. Automatic consumer video summarization by audio and visual analysis. IEEE International Conference on Multimedia and Expo. Барселона, Іспанія. 2011.

УДК 004.94

ЯСЕНОВА А.В.

ДИВЕРСИФІКАЦІЯ РИЗИКІВ НА РИНКУ ІНОЗЕМНИХ ВАЛЮТ

В даній статті розглянуто поняття диверсифікації, методи диверсифікації на ринку FOREX та їхні відмінності. Продемонстровано практичне значення і використання такої величини як drawdown на прикладі курсу валютної пари долар-євро. Також демонструється практичний розрахунок співвідношення активів в інвестиційному портфелі в залежності від значення обраної міри ризику.

ДИВЕРСИФІКАЦІЯ, ПРОСАДКА, РИЗИК, РИНОК ІНОЗЕМНИХ ВАЛЮТ, ГРАФІК Q-Q

The subject of the article is the diversification concept and methods in the FOREX market and their differences. The practical value and use of a drawdown value is demonstrated using EURUSD exchange rate example. This article also shows the mathematical calculation of the assets ratio in the investment portfolio, depending on the value of the selected risk measure.

DIVERSIFICATION, DRAWDOWN, RISK, FOREX EXCHANGE MARKET, Q-Q PLOT

1. Вступ

В даній статті розглянуто поняття диверсифікації, методи диверсифікації на ринку FOREX та їхні відмінності. Продемонстровано практичне значення і використання такої величини як drawdown на прикладі курсу валютної пари долар-євро. Також демонструється практичний розрахунок співвідношення активів в інвестиційному портфелі в залежності від значення обраної міри ризику.

2. Поняття ризику

Ринок іноземних валют відкриває можливості для розвитку дослідникам із багатьох галузей, адже має велику кількість відкритих даних для аналізу і безпосередньо поєднує в собі багато сфер людської діяльності. Тому форекс - це чудовий дослідницький майданчик як для економістів і аналітиків, так і для спеціалістів з аналізу даних чи програмістів (сервіси для збору даних і торгів на біржі). Разом з цим потрібно усвідомлювати, що більшість людей знайомляться з ринком іноземних валют для того, щоб отримувати прибуток. Зазвичай “новачки” не усвідомлюють своєї некомпетентності в даній сфері і не здатні в повній мірі оцінити можливі втрати під час торгів.

Для того, щоб скоротити свої збитки трейдеру необхідно вміло зважувати ризики тих активів, якими він оперує.

Ризики інвестиційної діяльності — це імовірність отримати відмінний від очікуваного результат інвестиційної діяльності внаслідок дії екзогенних і ендогенних чинників впливу. Під очікуваним результатом мається на увазі отримання доходів або/і досягнення певного соціального, технологічного, інформаційного, інноваційного та інших ефектів. [1]

3. Поняття диверсифікації

Для того, щоб скоротити збитки і компенсувати їх за рахунок наступного прибутку слід диверсифікувати свої інвестиції.

Диверсифікація (новолат. diversificatio — зміна, різноманітність; від лат. diversus — різний і facere — робити) — володіння найрізноманітнішими фінансовими активами, кожен з яких має різний рівень ризику, з метою зниження загального ступеня ризику портфеля в цілому. [2]

Принцип диверсифікації полягає в розподіленні грошових ресурсів між різними торговими інструментами. Це дозволяє мінімізувати ризики, які виникають при торгівлі чи/та інвестуванні без зменшення доходів. Велика ефективність за допомогою диверсифікації портфеля досягається, якщо додавати в нього негативно скорельовані активи. Якщо перейти від теорії до практичного прикладу, негативно скорельовані активи - це такі активи, які змінюються в ціні незалежно один від одного. Тобто, падіння ціни одного активу має компенсуватися зростанням вартості іншого. [3]

4. Приклад диверсифікації портфелю активів

Зазвичай інвестори вкладають кошти в два основні класи на ринку цінних паперів - акції та облігації. Окрім них є також золото, індекси, фонди. На ринку іноземних валют основними активами є валютні пари. Спектр можливих інструментів торгівлі дуже великий, тому обрати складові для торгового портфелю випадковим чином доволі складно.

Перш за все потрібно приділити увагу таким речам як: ризик, кореляція, прибутковість. Зауважимо, що головним з цих показників при диверсифікації є кореляція фінансових інструментів.

На графіку нижче показані курси валют австралійський долар (AUD), канадський долар (CAD), євро (EUR), швейцарський франк (CHF) і новозеландський долар (NZD) з червня по грудень 2018 року. Легко помітно, що деякі з них постійно або протягом певних проміжків часу сильно скорельовані. Наприклад, лінії курсів австралійського долара і новозеландського долара майже ідентичні. На початку досліджуваного періоду вони ростуть в ціні, а, наприклад, канадський долар і швейцарський франк падають. Проте, в жовтні австралійський долар і новозеландський долар почали падати в ціні, а канадський долар і швейцарський франк навпаки. Разом з цим курс євро залишився практично незмінним, тобто в загальному протягом досліджуваного періоду курс цієї валюти не корелював з іншими спостережуваними активами.

Якби інвестор намагався скласти торговий портфель з даних валютних пар, то для нього наведена вище інформація означала б наступне: в портфель необхідно додати негативно скорельовані валюти (в даному випадку хоча б дві: одна - це австралійський долар або новозеландський долар, інша - це канадський долар або швейцарський франк).

Також дуже важливо визначити співвідношення активів у портфелі. Наприклад, трейдер має $1000 і хоче торгувати двома парами. Можна розподілити доступні кошти між активами на рівні частини, проте більш розумним підходом є визначити певну величину, яка буде мірою ризикованості активу і обчислити її для кожного активу. А вже потім, в залежності від того, як співвідносяться обчислені значення, визначити співвідношення активів портфелю. Наприклад, такою величиною може виступати волатильність. Припустимо, що перша валюта має волатильність стд, а друга 4 стд. В такому випадку можна сказати, що перши актив є менш ризикованим, і більш оптимально вкласти в нього $1000 * (4 / 5) = $800, а в другий $1000 - $800 = $200.

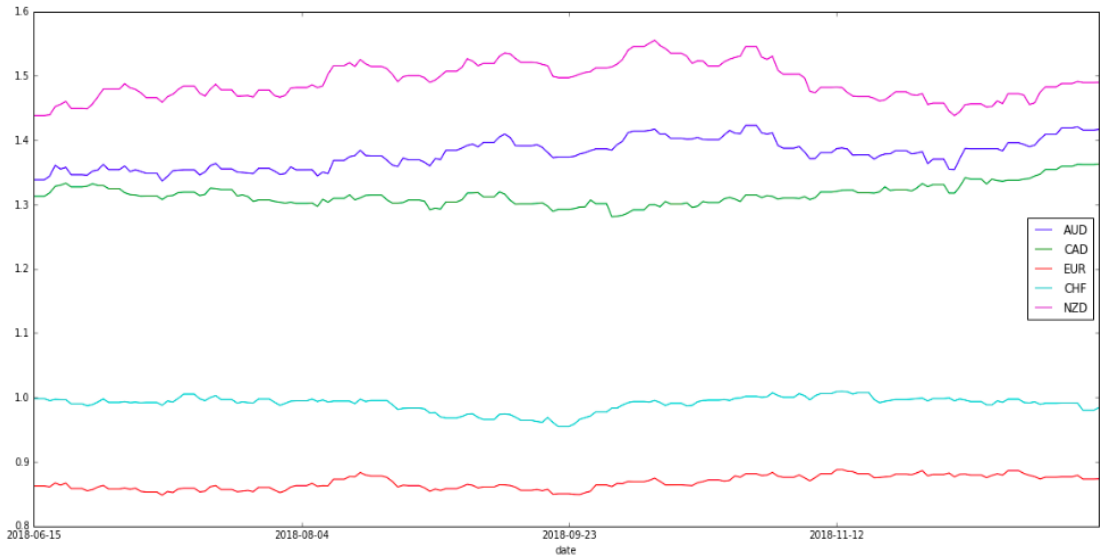


Рис.1. Курси валют відносно долара.

5. Поняття міри ризику. Value At Risk.Drawdown.

Зараз теорія оптимізації інвестиційних портфелів стрімко розвивається і є багато суперечок щодо того чи іншого способу мінімізації збитків. Одним із стандартних і найбільш поширених методів оцінки ризику є так званий VaR (Value At Risk — вартісна міра ризику), а найбільш очевидним показником ризикованості активів - просадка.

Value At Risk (VaR) — вартісна міра банківського ризику. Поширене загальноприйняте в усьому світі позначення «VaR». Це виражена в грошових одиницях оцінка величини, яку не перевищать очікувані протягом даного періоду часу втрати з заданою ймовірністю. [4]

Drawdown (просадка) - це відхилення балансу/вартості торгового акаунта/активу відносно максимального історичного (пікового) значення.

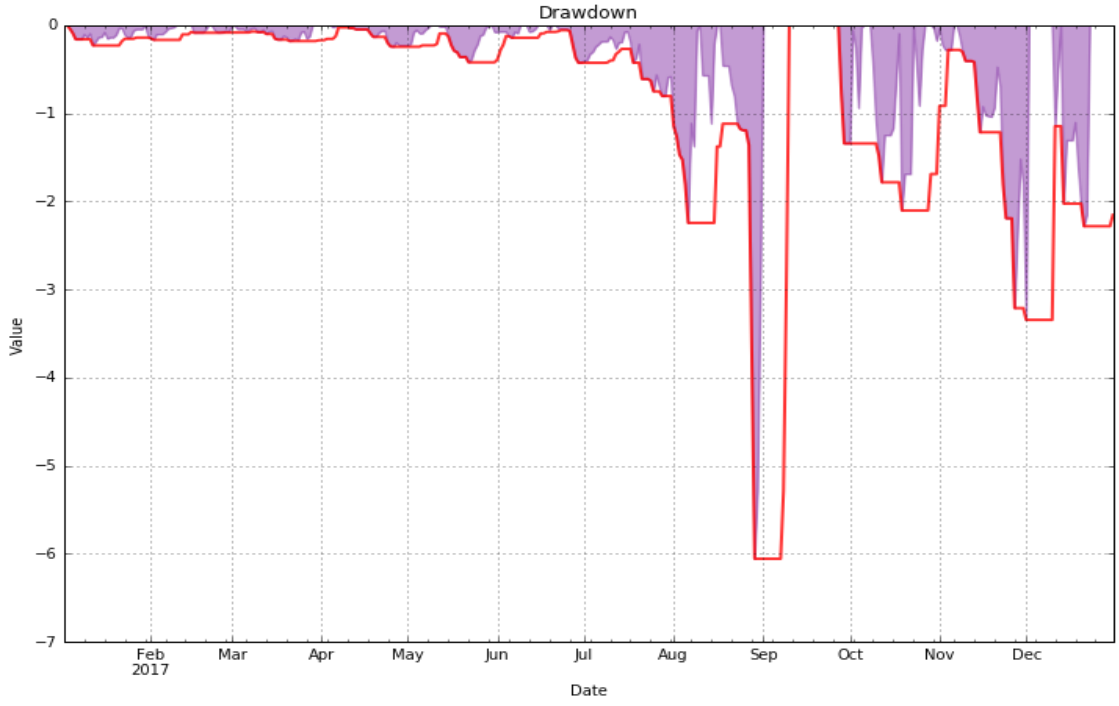


Рис.2. Просадки валютної пари долар-євро протягом 2017 року.

Неозброєним оком помітна велика просадка в серпні-вересні, також із загальної картини виділяється грудень.

Отримана інформація допомагає інтерпретувати коливання курсу валют:

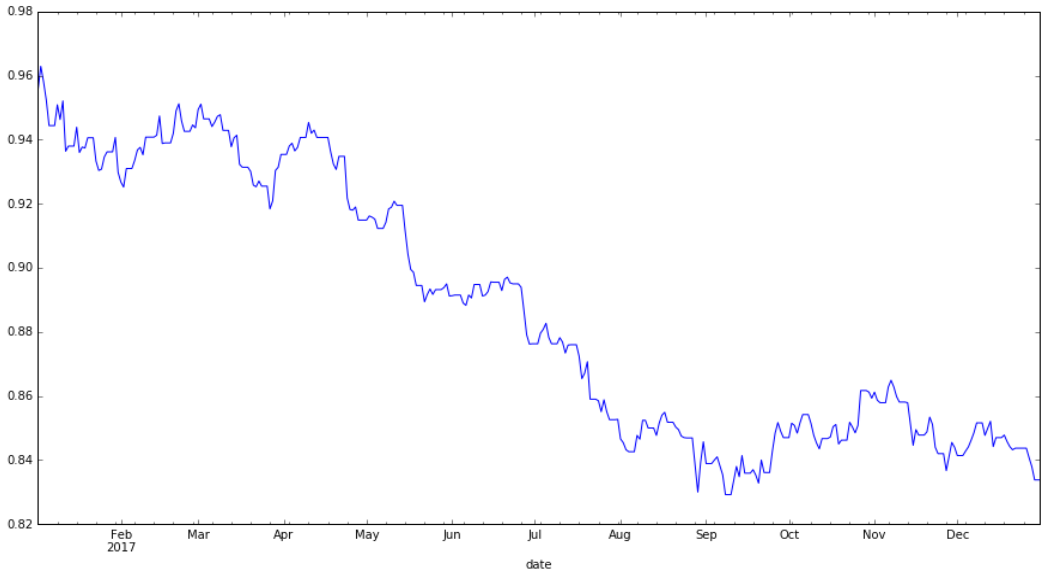


Рис.3. Курс валютної пари долар-євро протягом 2017 року.

З червня почалося зниження ціни, а в період з вересня по жовтень значення ціни було найнижчим. Після невеликого періоду відновлення в грудні ціна знову знизилася.

За допомогою просадок значно легше спостерігати значні відхилення в ціні. Ця величина інтерпретується легше, ніж звичне зображення курсу валюти.

7. Графік Q-Q

Менш поширеним і більш науковим методом диверсифікації є використання графіка Q-Q.

Графік Q-Q («Q» позначає квантиль) — імовірнісний графік у математичній статистиці, який являє собою графічний метод для порівняння двох розподілів ймовірностей, ставлячи їх квантилі один проти одного.Якщо два порівнюваних розподілів схожі, точки в графіці Q-Q будуть приблизно лежати на прямій y = x. Якщо розподіли лінійно пов'язані, точки в графіці Q-Q будуть приблизно лежати на одній прямій, але не обов'язково на прямий y = x. [5]

Для досліджуваного періоду часу (2017 рік) пари долар-євро обчислили відносні зміни ціни за день ((PR[i] - PR[i-1]) / PR[i-1], де PR - ціна в день i). Перевіримо припущення, що отриманий набір значень має нормальний розподіл.

Q-Q графік має форму дуги або «S» форму, це означає, що теоретичний розподіл менш асиметричний, ніж практичний і практичний розподіл має важчий хвіст, ніж теоретичний. цілому значення, досліджуване на практиці є досить близьким до теоретичного, тобто можемо стверджувати, що величини належать до одного сімейства розподілів.

Для диверсифікації важливо дослідити наскільки великий розкид хвостів практичного значення і як сильно значення, досліджуване на практиці відхиляється від теоретичного. Чим більш схожі ці дві величини, тим менш ризикованим є досліджуваний актив.

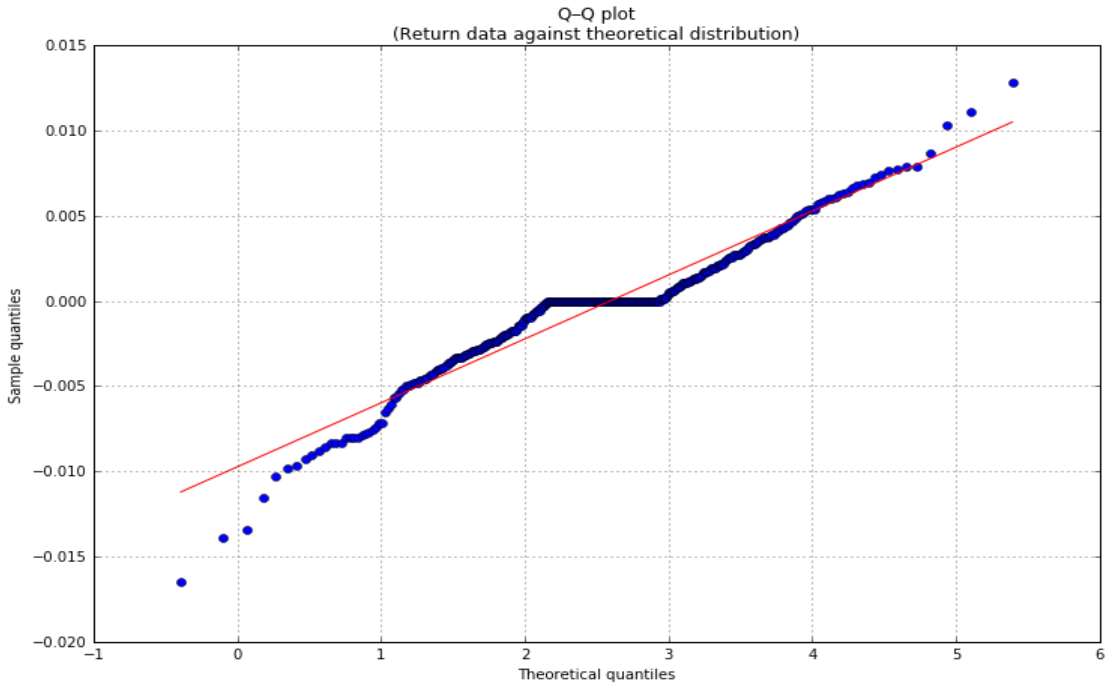


Рис.4. Графік Q-Q для відносних змін ціни валютної пари долар-євро протягом 2017 року

Висновок

Отже, будь-який, навіть найбільш успішний бізнес, не може завжди приносити прибуток. Тому є метод, який збільшує стійкість портфелю і максимально зменшує ризик в критичних ситуаціях. Диверсифікація - це спосіб управління можливими збитками портфеля, що складається з різних негативно скорельованих фінансових інструментів. [3] Інвестиційний портфель не застрахує від короткочасних збитків. Проте, якщо до його складу входять негативно скорельовані активи різних класів, то загальний інвестиційний ризик мінімізується без зниження прибутковості.

Список літератури

1. Ризики інвестиційної діяльності [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://uk.wikipedia.org.
2. Диверсифікація [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://uk.wikipedia.org.
3. Диверсифікація [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://freshforex.org/encyclopedia-forex/diversification/.
4. Value At Risk (VaR) [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://uk.wikipedia.org.
5. Графік Q-Q [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://uk.wikipedia.org.

УДК 004.05

БУРЛАЧЕНКО Є.О.

ПРАКТИЧНЕ ЗАСТОСУВАННЯ СИСТЕМ АВТОМАТИЗАЦІЇ ІНТЕГРАЦІЇ ТА ДОСТАВКИ ПРОГРАМНИХ ПРОДУКТІВ

У даній статті розглянуто застосування систем автоматизації інтеграції та доставки програмних продуктів. Демонструється збільшення швидкодії, надійності та доступності системи. Наведено приклад розрахунку необхідної кількості копії програмного забезпечення для найкращої роботи системи.

ІНТЕГРАЦІЯ, ДОСТАВКА, МАСШТАБУВАННЯ, ВИСОКА ДОСТУПНІСТЬ

The subject of this article is an application of systems for automation of integration and delivery of software products. It demonstrates an increase in the speed, reliability and availability of the system. An example calculation of the required amount of application processes for the best system efficiency is given.

CONTINUOUS INTEGRATION, CONTINUOUS DELIVERY, SCALING, HIGH AVAILABILITY

1.Вступ

Інтеграція - процес злиття гілок програмного коду продукту та виконання збірки разом з тестуванням, завдяки чому можна заздалегідь виявити проблеми їх взаємодії.

Неперервна інтеграція - автоматизація процесу інтеграції для проведення збірок після кожного оновлення кодової бази продукту розробниками.

Доставка - процес завантаження та запуску нової версії програмного продукту до цільової платформи. У випадку з веб сервісом - розгортання нової версії сервісу на сервері з забезпеченням обслуговування користувачів попередньої до останнього запиту; Для самостійного продукту - процес завантаження його до центрального репозиторію (Apple App Store, Google Play market, Playstation Network, Steam, власна розробка тощо) для того щоб користувачі мали змогу якнайшвидше отримати нову версію.

Неперервна доставка - автоматизація процесу доставки для якнайшвидшого її виконання після кожного отримання робочої збірки продукту.

Масштабування - зміна кількості одночасно робочих копій ПЗ в залежності від навантаження на систему, що дозволяє досягти найкращого співвідношення доступності системи до вартості її експлуатації.

Завдяки цим двом методикам компанії можуть збільшити прибуток за рахунок зменшення часу очікування користувачем нових версій, зменшення навантаження на робітників для виконання однотипних дій, збільшення якості продукту завдяки автоматизованим тестам що не пропустять поганий продукт для доставки.

Одним із сценаріїв побудови таких процесів є сценарій взаємодії сервісів GitHub, CircleCI та Google Cloud Platform, а також таких технологій як Docker та Kubernetes.

GitHub - сервіс для збереження програмного коду продукту, яких дозволяє працювати з його версіями колективно, підтримує інтеграцію зі сторонніми ресурсами. Завдяки цій підтримці будь який код що завантажується на даний сервіс може бути неперервно інтегрований.

2. Неперервна інтеграція та доставка

CircleCI - хмарний сервіс для неперервної інтеграції та неперервної доставки. Він підтримує інтеграцію с GitHub, а також надає можливість користуватися Linux та macOS середовищами для збірки ПЗ. CircleCI використовує контейнери Docker в якості робітників для збірок, завдяки чому відбувається тісна інтеграція з продуктами що постачаються у вигляді Docker контейнерів, а також гнучкість в обслуговуванні - сервісу не потрібно тримати окрему віртуальну або фізичну

машину для кожного клієнта, достатньо лише запускати окремий контейнер з робітником для кожного процесу збірки. Також CircleCI підтримує взаємодію с сервісами Google Cloud Platform, а саме, для даного сценарію, з Google Container Registry - сервіс для хмарного зберігання зліпків контейнерів Docker.

Google Cloud Platform - хмарний сервіс, що надає доступ до ряду послуг по використанню обчислювальних можливостей, зберігання даних, організації хмарних мереж тощо. Сервіс який використовується для даного сценарію - Kubernetes - це контейнерний оркестратор. Він забезпечує безперервну роботу контейнеризованого ПЗ у кластері машин, а також їх взаємодію. Він об'єднує фізичні або віртуальні машини у кластер, на якому, завдяки вбудованому планувальнику рівномірно запускає контейнери. У термінах Kubernetes є поняття розгорнення - один чи більше контейнерів з однаковим ПЗ що автоматично встановлюється у кластер завдяки декларативному сценарію описання - Kubernetes маніфесту. Копії ПЗ у розгортанні, або, у термінах Kubernetes - поди, розподіляються по різним машинам кластеру для досягнення максимальної стійкості до відмов та розподілення навантаження на систему. Але найбільш цікавим побічним ефектом наявності декількох копій ПЗ є можливість проводити неперервну доставку - поки користувачі старої версії ПЗ закінчують виконувати запити, поди з новою версію запускаються автоматично, без участі адміністратора. Kubernetes зберігає версії релізів розгортання, тому, за необхідності (наприклад нова версія виявиться нестабільною) , він може автоматично повернути розгортання до минулого стану. Ця гнучкість дозволяє виконувати доставку продукту багато разів за малий проміжок часу, не біючись можливості зламати систему та отримати довгий період простою (англ. Downtime) системи, що може призвести до великих збитків.

Docker - ПЗ для автоматизації встановлення та управління ПЗ у середовищах з підтримкою контейнеризації. Завдяки Docker можливо зручно створювати, запускати, переміщувати та будувати взаємодію контейнерів ПЗ.

Контейнеризація - системний вид віртуалізації, що полягає у ізоляції ПЗ та доступних йому ресурсів на рівні ядра ОС, даючи можливість запускати ПЗ та необхідні йому компоненти у окремому стандартизованому середовищі, що не залежить від операційної системи та її апаратного забезпечення. Усі файли необхідні для роботи ПЗ наявні у файловій системі контейнера, завдяки чому він може бути використаний повторно та легко встановлюється. Один й той самий контейнер може бути використано у середовищі для розробки, тестування, релізу тощо.

CircleCI також може бути використаний для неперервної доставки. Але, для неперервної інтеграції та доставки з використанням Github та Kubernetes краще використати сервіс Clipper.

3. Масштабування

Наступною проблемою доставки є масштабування: За наявності надто малої кількості копій елементів системи (серверів) можлива велика кількість відмов в обслуговуванні, надто велика кількість призводить до простою та неповної експлуатації ресурсів, що є надлишковим використанням коштів на утримання ресурсів (серверних платформ, хмарних сервісів тощо). Для розрахунку необхідної кількості додаткових копій (серверів) ПЗ використовується наступна формула

$$Z_x = \frac{n * (Z - Z_1)}{Z_1} (1)$$

Де n - поточна кількість серверів, Z- поточна завантаженість серверу , Z_1 - максимально допустима завантаженість серверу. Завантаженість серверу можна рахувати в будь яких зручних одиницях одного типу, наприклад, мілісекундах. Так як кількість серверів - ціле число, результат потрібно округлити до найближчого цілого. Наприклад,

максимально допустимий час відповіді серверу на запит, що вважається ефективним є 4117 мілісекунд, поточний середній час

відповіді - 5085 мс, кількість серверів у системі - 3.

$Z = 5085$ мс , $Z_1 = 4117$ мс , $n = 3$

$$Z_x = \frac{3 * (5085 - 4117)}{4117} = 0.71$$

Округляємо до найближчого цілого - 1, тобто система є перезавантаженою, та для досягнення найкращої ефективності системи потрібно додати ще один сервер;

Максимально допустимий час відповіді серверу на запит, що вважається ефективним є 5000 мілісекунд, поточний середній час відповіді - 1024 мс, кількість серверів у системі - 4.

$$Z_x = \frac{4 * (5000 - 1024)}{1024} = -3.18$$

округляємо до найближчого цілого -3, тобто навантаження системи достатньо низьке, щоб з неї можна було вивести 3 сервери для економії коштів.

Для досягнення максимальної ефективності дані розрахунки потрібні виконуватися автоматично, як і сам процес масштабування - у системах з великою кількістю користувачів можливі миттєві злети навантаження, що потребують швидкої реакції з правильним масштабуванням. Менш економне використання ресурсів є допустимим, тоді як відмови під час пікових навантажень можуть призвести до значної втрати прибутку бізнесом через незадоволеність клієнтів. Завдяки інтеграції Clipper з Kubernetes, що в свою чергу використовує сервери платформи Google Cloud, масштабування відбувається автоматично - при збільшенні навантаження kubernetes сам обирає скільки додаткових копій ПЗ потрібно запустити на одному з серверів кластеру. Якщо ж серверів кластеру недостатньо для ефективної роботи системи, kubernetes виконує запит до Google Cloud на додання нового серверу у кластер. Те ж саме виконується у зворотному випадку, для економії коштів під час низького навантаження на систему. Якщо за якихось причин потрібне ручне керування кількістю копій ПЗ, Clipper надає таку можливість на сторінці “deployment”.

Результати масштабування

На наступному графіку зображено кількість запитів на секунду що обробляє тестове ПЗ при симуляції одночасної роботи 50 користувачів при одній, двох та трьох копіях ПЗ відповідно

Збільшення копій ПЗ призводить до збільшення максимальної кількості оброблених запитів за секунду, за рахунок чого також зменшується затримка на відповідь та тривалість обробки індивідуальних запитів.

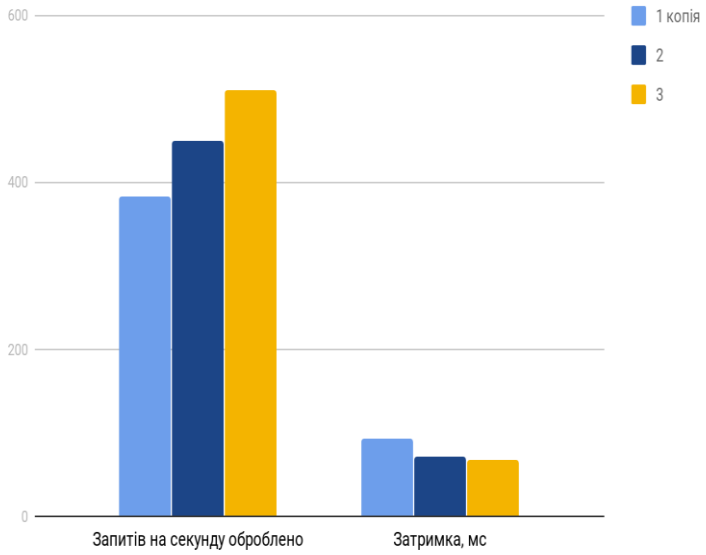


Рис. 1. Графік доступності системи

Висновки

Сервіси по автоматизації інтеграції та доставки ПЗ є необхідними для швидкої розробки та контролю їх якості, завдяки виконанню однотипних але необхідних дій максимально швидко користувачі продукту та його розробники можуть безперебійно отримувати нові версії продукту до декількох разів на день, а серверна інфраструктура автоматично пристосовується до навантаження, стаючи максимально швидкою та доступною для користувачів та дешевою для розробників.

Список літератури

1. Arachchi. Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management / Arachchi, P. Indika, S.A.I.B.S.., 201.

УДК 004.067
СОЛОДКИЙ А.В.

ПРОГРАМНО-ТЕХНІЧНИЙ КОМПЛЕКС ДЛЯ ЕКСТРЕНОГО РЕАГУВАННЯ НА РУХ ЛЮДЕЙ У ВІДЕОПОТОЦІ

В статті описаний спосіб організації системи спостереження яка базується на роботі алгоритмів глибокого навчання. Коротко описано метод роботи згорткової нейронної мережі. Було описано постановку задачі та опис рішення на прикладі системи реагування на рух людей у відеопотоці.

Ключові слова: спостереження, відеокамери, глибоке навчання, екстрене сповіщення, виокремлення, класифікація

The article describes a way of organizing a monitoring system based on the work of deep learning algorithms. A method for working with a convolutional neural network is briefly described. The task statement and a description of the solution were described on the example of a system for responding to the movement of people in a video stream.

Keywords: surveillance, cameras, in-depth training, emergency notification, identification, classification.

1. Вступ

Відеонагляд є невід'ємною частиною заходів забезпечення безпеки та спостереження. Здебільшого ця робота передбачає тривалі періоди спостереження за відеопотоком у пошуках, можливо, декількох секунд шуканої події. Завдяки сучасним технологіям цю проблему можна вирішити за допомогою використання глибинного навчання. При аналізі сучасних алгоритмів можна сказати, що уже існуючі класифікатори зображень перевершили точку людського рівня. Машина може здійснювати спостереження за об'єктами на кращому рівні, в порівнянні з людиною. Також, за допомогою автоматизованого аналізу відеооб'єктів можливе екстренне реагування на нетиповий рух у відеопотоці.

Дана задача є задачею розпізнавання об'єктів на відео. Існує ряд алгоритмів які забезпечують виокремлення об'єктів із зображень та відеорядів. Для того, щоб імітувати людський рівень можливості класифікації об'єктів, вчені розбили задачі зорового сприйняття на чотири категорії: класифікація, локалізація, об'єкт виявлення та сегментація зображення.

В даний час найкращу якість показують алгоритми, засновані на використанні згорткових нейронних мереж (Convolutional Neural Network, CNN). Відмінності алгоритмів даної групи, як правило, виявляються в архітектурі мереж і параметрах їх навчання [1,

2]. Також можуть відрізнятися методи обробки ознак [3, 4], а також підходи до вирішення проблеми пошуку об'єктів різного масштабу [5].

За основу архітектури моделі буде взяте рішення під назвою Faster R-CNN [6]. Faster R-CNN зараз є канонічною моделлю для визначення об'єктів на основі глибокого навчання.

2. Постановка задачі

Будемо використовувати наступну модель задачі визначення об'єктів:

Ω – множина об'єктів розпізнавання (простір образів).

ω : π ∈ Ω – об'єкт розпізнавання (образ).

g(ω) : Ω → M, M = {1,2,...,m} – індикаторна функція, що розбиває простір образів Ω на m класів, що не перетинаються – Ω1, Ω2,...,Ωm.

Індикаторна функція невідома спостерігачу.

X – простір спостережень, які сприймає спостерігач (простір ознак).

x(ω) : Ω → X – функція, яка ставить у відповідність кожному об'єкту ω точку x(ω) в просторі ознак. Вектор x(ω) – це образ об'єкта, який сприймає користувач.

В просторі ознак визначені множини точок, що не перетинаються – Ki⊂X, i = 1,2..., m, що відповідають образам одного класу.

$\hat{g}(x) : X \to M$ – вирішальне правило – оцінка для $g(\omega)$ на основі x(ω), тобто $\hat{g}(x) = \hat{g}(x(\omega))$.

Нехай xj = x(ωj), j = 1,2..., N – доступна спостерігачу інформація про функції $g(\omega)$ і x(ω), але самі ці функції спостерігачу невідомі. Тоді (gj, xj), j = 1,2..., N – множина прецедентів.

Задача заключається в побудові такого вирішального правила $\hat{g}(x)$, щоб розпізнавання проводилось з мінімальною кількістю помилок.

3. Метод, заснований на повністю згортковій мережі з малою кількістю вагів

У більшості алгоритмів визначення об'єктів застосовуються дуже глибокі згорткові нейронні мережі, які мають велику кількість ваг (AlexNet [7], VGG-16 [8], ResNet [9]). Метод FastCNN [10], доводить, що це завдання можна вирішити за допомогою згорткової нейронної мережі зі значно меншою кількістю шарів і ваг.

Спочатку навчається мережа, яка вирішує задачу бінарної класифікації «особа / не особа». Мережа приймає на вхід трьохканальний зображення у форматі RGB з роздільною здатністю 32×32 пікселя. Потім зображення послідовно обробляється за допомогою семи згорткових шарів. Після кожного згорткового шару застосовується активаційна функція PReLu [11], яка дозволяє робити нелінійні перетворення з виходом попереднього шару і в той же час швидко обчислювати похідну. Вихід мережі після кожного згорткового шару має менший розмір, ця властивість дозволяє вирішувати задачу класифікації без застосування шарів іншого типу. Виходом останнього згортокового шару є пара чисел, кожне з яких визначає достовірність приналежності до одного з двох класів. Мережа з такою архітектурою має приблизно в 800 разів менше ваги, ніж мережа AlexNet. Таке значне зменшення кількості ваг дозволяє набагато скоротити час роботи детектора, а також скоротити час, необхідний для навчання мережі. Так як описана мережа є повністю згортоковою, то вона може бути застосована до зображень довільного розміру. В результаті всіх обчислень на виході мережі виходять дві матриці, які описують достовірності приналежності конкретної області зображення до одного з класів.

4. Метод, на основі алгоритму Faster R-CNN

Алгоритм [6] складається з двох модулів: повністю згорткової нейронної мережі, яка використовується для вилучення ознак із зображення і знаходження передбачуваних прямокутників, і детектора на основі R-CNN [12], який використовує ці прямокутники. Зазначені модулі об'єднуються в одну нейронну мережу Faster R-CNN (рисунок 1).

На першому етапі зображення довільного розміру подається на вхід нейронної мережі, яка забезпечує виділення прямокутників для подальшої класифікації Region Proposal Network (RPN). Ця мережа є повністю згорковою нейронною мережею, яка отримана з мережі VGG-16 шляхом видалення останніх повністю зв'язкових шарів і додаванням одного або декількох повністю згорткових шарів. Додані згорткові шари використовуються безпосередньо для визначення передбачуваних прямокутників. Додані шари обходяться ковзаючим вікном по карті ознак, отриманої з мережі VGG-16, і для кожної позиції вікна витягується вектор ознак малої розмірності. Вектори ознак подаються на вхід двох нових повністю зв'язковим шарам. Один з цих шарів використовується для уточнення меж прямокутника, а інший для класифікації об'єкта, розташованого усередині цього прямокутника.

У кожній позиції ковзаючого вікна одночасно можуть розглядатися кілька прямокутників, які володіють різним розміром або різним співвідношенням сторін. За допомогою такого підходу на зображенні одного масштабу можна знайти особи різної величини.

Після того як передбачувані прямокутники, що містять об’єкти, отримані, їх необхідно класифікувати. Зазвичай для цього використовується інша мережа, яка навчається окремо. Але в даному випадку пропонується використовувати для класифікації частина шарів з мережі RPN. Це дозволяє значно скоротити час визначення, а також спрощує процес навчання моделі, так як модель являє собою єдину нейромережу, яку можна навчати за допомогою методу зворотного поширення помилки (Back Propagation). Після застосування мережі на виході виходить набір прямокутників. Кожній особі на зображенні можуть відповідати кілька різних прямокутників. Тому завершальним етапом визначення є об’єднання прямокутників.

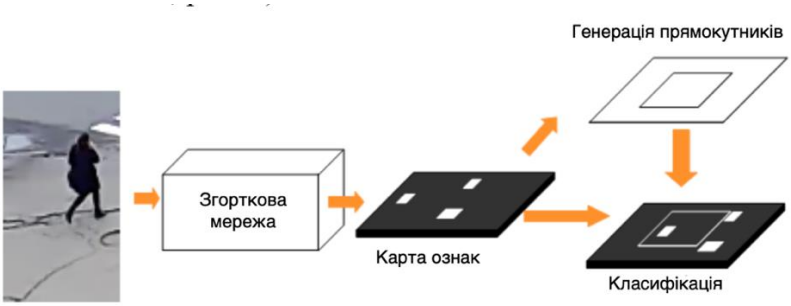


Рис. 1. Схема роботи Faster R-CNN

5. Вдосконалення алгоритму

Архітектуру Faster R-CNN можна удосконалити шляхом додавання ще однієї гілки, яка передбачає положення маски, що покриває знайдений об’єкт, і, таким чином вирішує завдання instance segmentation. Маска являє собою просто прямокутну матрицю, в якій 1 на деякій позиції означає приналежність відповідного пікселя об'єкту заданого класу, а 0 – що піксель об’єкту не належить.

Отриману архітектуру можна умовно поділити на CNN-мережу обчислення ознак зображення і об’єднання частин, що відповідають за пророкування охоплючої рамки, класифікацію об’єкта і визначення його маски.

Виділення маски відбувається таким чином: маски передбачаються окремо для кожного класу, без попереднього знання того, що саме зображено в регіоні, і потім просто вибирається маска класу, який переміг в незалежному класифікаторі.

Одна з основних модифікацій, що виникли через необхідність передбачати маску – зміна процедури RoIPool (обчислює матрицю ознак для регіону-кандидата) на так звану RoIAlign. Справа в тому, що карта ознак, отримана з CNN, має менший розмір, ніж вихідне зображення, і регіон, який охоплює на зображенні цілочисельну кількість пікселів, не виходить відобразити в пропорційний регіон карти з цілочисельною кількістю ознак.

У RoIPool проблема вирішувалася просто округленням дрібних значень до цілих. Такий підхід нормально працює при виділенні охоплює рамки, але обчислена на основі таких даних маска виходить занадто неточною.

На противагу цьому, в RoIAlign не використовується округлення, всі числа залишаються дійсними, а для обчислення значень ознак використовується білінійная інтерполяція по чотирьом найближчим цілочисельним точкам.

Різницю можна показати схемою, що зображена на рисунку 3. Тут штрихованої сіткою позначена карта ознак, а безперервною – відображення на карту ознак регіону-кандидата з початкової фотографії. В даний регіон має потрапити 4 групи по 4 ознаки, позначених на малюнку точками. На відміну від процедури RoIPool, яка за рахунок округлення просто б вирівняла регіон по цілочисельним координатами, RoIAlign залишає точки в їх поточних місцях, але обчислює значення кожної з них за допомогою білінійної інтерполяції по чотирьом найближчим ознаками.

В таблиці 1 наведено результат при використанні модифікованого алгоритму та при використанні Faster R-CNN. Мірою будемо використовувати середню точність, яка обчислює середнє значення точності для відношення усіх позитивних результатів до усіх можливих відповідей.

Таблиця 1: Результат експерименту

Метод	Середня точність
Faster R-CNN	0.704
Даний метод	0.749

Як видно з результатів експерименту, використання модифікованого методу, покращує середню точність завдяки вдосконаленому процесу передбачення і сегментації.

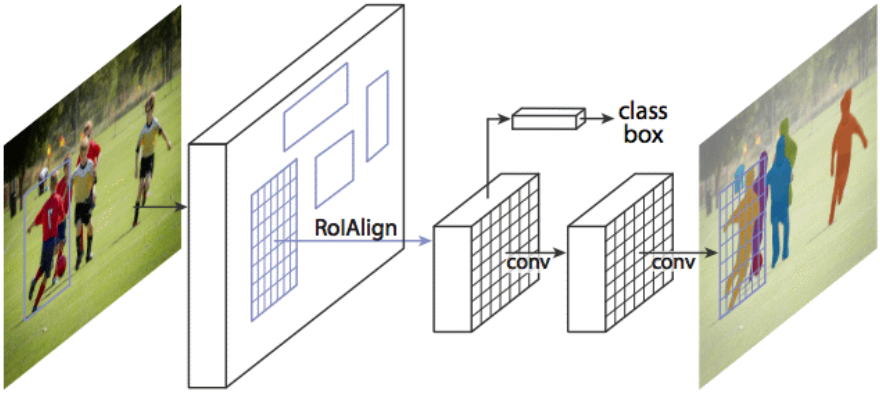


Рис. 2. Схема роботи удосконаленого алгоритму Faster R-CNN

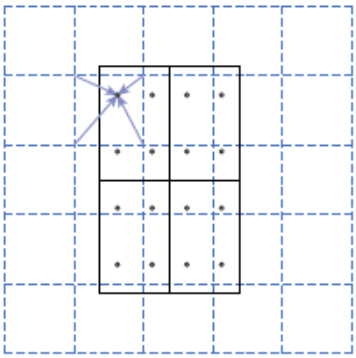


Рис. 3. Схема роботи RoIAlign

Список літератури

1. Sun X., Wu P., Hoi S. Face Detection using Deep Learning: An Improved Faster RCNN Approach [Електронний ресурс] // Режим доступу: https://arxiv.org/abs/1701.08289
2. Jiang H., Learned-Miller E. Face Detection with the Faster R-CNN [Електронний ресурс] // Режим доступу: https://arxiv.org/abs/1606.03473
3. Farfade S., Saberian M., Li L. Multi-view Face Detection Using Deep Convolutional Neural Networks [Електронний ресурс] // Режим доступу: https://arxiv.org/abs/1502.02766
4. Ranjan R., Patel V.M., Chellappa R. A Deep Pyramid Deformable Part Model for Face Detection [Електронний ресурс] // Режим доступу: http://arxiv.org/abs/1508.04389
5. Hu P., Ramanan D. Finding Tiny Faces [Електронний ресурс] // Режим доступу: https://arxiv.org/abs/1612.04402
6. Faster R-CNN [Електронний ресурс] // Режим доступу: https://arxiv.org/abs/1506.01497
7. Krizhevsky A., Sutskever I., Hinton G. ImageNet classification with deep convolutional neural networks // Advances in Neural Information Processing Systems. – 2012.
8. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition [Електронний ресурс] // Режим доступу: https://arxiv.org/abs/1409.1556
9. He K., et al. Deep Residual Learning for Image Recognition[Електронний ресурс] // Режим доступу: https://arxiv.org/abs/1512.03385

10. Triantafyllidou D., Tefas A. A Fast Deep Convolutional Neural Network for Face Detection in Big Visual Data [Електронний ресурс] // Режим доступу: http://arxiv.org/abs/1508.04389
11. He K., et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification [Електронний ресурс] // Режим доступу: https://arxiv.org/abs/1502.01852
12. Caffe [Електронний ресурс] // Режим доступу: http://caffe.berkeleyvision.org

УДК 519.876.5

СКОРИК В.А.
ЖДАНОВА О.Г.

АНАЛІЗ МОЖЛИВОСТЕЙ ЗЛОВЖИВАНЬ ЗІ СТОРОНИ КОРИСТУВАЧІВ В ДЕЦЕНТРАЛІЗОВАНИХ МЕРЕЖАХ З РОЗПОДІЛОМ НАГОРОД

У статті досліджується проблема зловживань зі сторони спільноти free-ride користувачів в децентралізованих мережах з розподілом нагороди. Пропонується нова структура для об'єднання free-ride користувачів у спільноту. Для даної структури досліджено вплив кількості зв'язків між користувачами, що належать спільноті free-ride користувачів, і іншими користувачами мережі на величину отримуваної спільнотою винагороди в залежності від параметрів побудови даної структури.

Ключові слова: децентралізовані мережі, P2P, зловживання, free-ride користувачі, розподіл нагороди.

The article investigates the problem of abuse by the community of free-ride users in decentralized networks with reward distribution. A new structure for uniting free-ride users into the community is considered. For this structure, the impact on the amount of the community received reward for the number of links between users belonging to the community of free-ride users and other users of the network, depending on the parameters of the construction of this structure is investigated.

Key words: decentralized networks, P2P, abuse, free-ride users, reward distribution.

Вступ

В даній статті продовжено дослідження поведінки спільнот free-ride користувачів в децентралізованих (P2P) мережах з розподілом нагород, яке було розпочате в роботі [1]. В попередньому дослідженні було запропоновано такі структури для об'єднання free-ride користувачів: *Ланцюг*, *Кільце*, *Зірка*, *Узагальнене кільце*. В результаті проведених експериментів було зроблено висновок, що об'єднання користувачів в структуру під назвою «Узагальнене кільце» є найбільш вигідним для спільноти free-ride користувачів з точки зору отримуваної ними нагороди. В даній роботі пропонується нова структура спільноти free-ride користувачів, а також аналізуються її властивості в залежності від зміни параметрів її побудови, параметрів алгоритму розподілу нагороди, кількості зв'язків з іншими користувачами мережі, які не належать спільноті free-ride користувачів.

Модель P2P мережі

В даному дослідженні, використовується модель запропонованої P2P мережі, описана за допомогою теорії графів в роботі [1]. В таблиці 1 наведені основні позначення, які використовуються в даній роботі.

Будемо використовувати наступні поняття: *основною мережею* будемо називати множину вузлів, які не належать спільноті free-ride користувачів, що розглядається; *кластером* будемо називати множину вузлів, що належать спільноті free-ride користувачів.

Структура кластеру користувачів

В даній роботі пропонується нова структура кластеру користувачів, яка далі називається *Другим узагальненим кільцем (УК2)*. Ця структура є більш загальним випадком *Узагальненого кільця*, структура якого описана в [1]. Узагальнення полягає в тому, що в даній структурі на кожному шарі з авторів розміщується не 1 вузол, а r. В цьому разі зменшується кількість вузлів, які знаходяться на кожному шарі з підписниками:

$$t = \frac{c}{m} - r$$

Табл. 1. Умовні позначення

Множина	Значення
A	Множина вершин графа, які відповідають авторам
S	Множина вершин графа, які відповідають підписникам
V	Множина всіх вершин графа $V=A\cup S$, $A\cap S=\varnothing$
V_1	Множина вершин, які належать поточній хвилі на певній ітерації в процесі розподілу нагород
V_2	Множина вершин, які належать наступній хвилі на певній ітерації в процесі розподілу нагород
E	Множина ребер графа. Для кожного ребра $(v_i, v_j)\in E$ виконується: $v_i\in S$, $v_j\in A$ (або навпаки)
Число	**Значення**
c	Кількість вершин, які входять до спільноти free-ride користувачів
h	Кількість нагороди спільноти free-ride користувачів після введення нагороди в усі вершини з множини S по одному разу
m	Кількість шарів в *Узагальненому кільці* або *Другому узагальненому кільці*
α_v	Частина від нагороди z_v, яка залишається в вершині v

Використання зовнішніх зв'язків

В деяких випадках перед спільнотою free-ride користувачів може постати питання утворення зв'язків з іншими користувачами мережі, які не належать даній спільноті.

Основними причинами використання кластером free-ride користувачів зв'язків з основною мережею можуть виступати:

— збільшення складності виявлення кластеру в структурі мережі; якщо кластер не має зв'язків з основною мережею, то він виступає окремою компонентою зв'язності і тому може бути виявлений за допомогою досить простого алгоритму [2];

— збільшення кількості отримуваної кластером винагороди;

Задача побудови зв'язків між спільнотою free-ride користувачів і основною частиною мережі є достатньо нетривіальною. Це зумовлено наступними складнощами:

1. спільнота free-ride користувачів повинна мати повну інформацію щодо структури мережі (всіх зв'язках між її учасниками), а також параметрів алгоритму розподілу нагороди; кількість доступної інформації регулюється правилами конкретної мережі і зазвичай є обмеженою;
2. зв'язки в мережі постійно змінюються: постійно виникають нові або руйнуються деякі старі, побудувавши зв'язки один раз, не можна гарантувати, що в певний момент часу, кількість отримуваної нагороди раптово не зменшиться.

Від того, яку інформацію про структуру мережі може отримати free-ride спільнота, залежить, чи буде існувати спосіб побудови зв'язків з основною частиною мережі, який би збільшував кількість отримуваної спільнотою нагороди.

В даному дослідженні припускається, що користувач може отримати інформацію лише про те, скільки зв'язків має той чи інший користувач, який виступає автором контенту.

Визначення найбільш перспективних зовнішніх зв'язків

При створенні зв'язків з основною мережею в умовах описаної раніше обмеженості наявної інформації спільнота free-ride користувачів не може розрахувати кількість нагороди, яка буде надходити до неї через ці зв'язки. В даній роботі використовується наступна стратегія побудови зв'язків від вузлів кластера, які відносяться до множини S, до вузлів з основної мережі, які належать множені A: будемо будувати зв'язки до авторів, які мають найменшу кількість підписників, при чому до одного автора будемо будувати не більше одного зв'язку від кластеру. Ідеальним випадком за цим алгоритмом вважається створення зв'язку з таким автором з основної мережі, який має лише одного підписника, який в свою чергу підписаний лише на цього автора.

Використання підходу максимізації PageRank

Спробуємо мінімізувати кількість нагороди, яка втрачається кластером через створені зовнішні зв'язки. Для цього використаємо ідею, яка схожа на підхід по максимізації PageRank, що описаний в роботі [3]: якщо кількість зв'язків користувача зі спільноти free-ride користувачів з користувачами з основної мережі буде значно меншою, ніж кількість зв'язків даного користувача з іншими користувачами даної спільноти, то частина від тієї нагороди, що надійде до цього користувача, яка буде втрачатися через його зовнішні зв'язки, буде незначною, а саме:

$$\frac{t_{out}}{t_{in}+t_{out}},$$

де t_{in} – кількість зв'язків з користувачами, що належать free-ride спільноті і належать V_2;

t_{out} – кількість зв'язків з користувачами з основної мережі, які належать V_2.

Структура основної частини мережі

В якості структури основної мережі в даному дослідженні використовується *мережа з переважним приєднанням* (PA мережа). Головним припущенням, яке використовується при побудові PA мереж, є те, що нові зв'язки утворюються з більшою імовірністю між вузлами, які вже мають велику кількість зв'язків [4]. Використання даного припущення робить даний вид мереж досить підходящим для моделювання реальних соціальних мереж.

Результати експериментів

Під час проведення експериментів в даній роботі використовується основна мережа з 5000 вузлів, з яких 300 вузлів належать множині A.

Проведемо експерименти з *Другим узагальненим кільцем*, яке складається з 60 вузлів (c) і має параметри $m=6, r=4$ або $m=4, r=6$.

Перевіримо, як змінюється кількість отримуваної кластером нагороди, в залежності від його параметрів і кількості зв'язків з основною мережею. Спочатку зафіксуємо параметри алгоритму розподілу нагороди $\alpha_1=0.1, \alpha_2=0.8$, потім встановимо параметри розподілу нагороди $\alpha_1=0.1, \alpha_2=0.5$ і повторимо експерименти. Перевіримо, скільки нагороди (h) буде отримувати УК2 з різними параметрами при використанні в якості основної – PA мережі. Результати експериментів представлені на рисунках 1, 2. Як бачимо, для структури, в якій кожний з підписників кластеру зв'язаний більшою кількістю внутрішніх зв'язків з вузлами кластеру, кількість нагороди кластеру більша при існуванні однакової кількості зв'язків від кожного підписника до вузлів з основної мережі. Також, можна побачити, що хоча при деяких параметрах алгоритму розподілу нагороди кількість нагороди кластеру спочатку зменшується при збільшенні кількості зв'язків від кожного з вузлів-підписників, що йому належать, до основної мережі, при подальшому збільшенні кількості цих зв'язків кількість нагороди кластеру починає монотонно зростати.

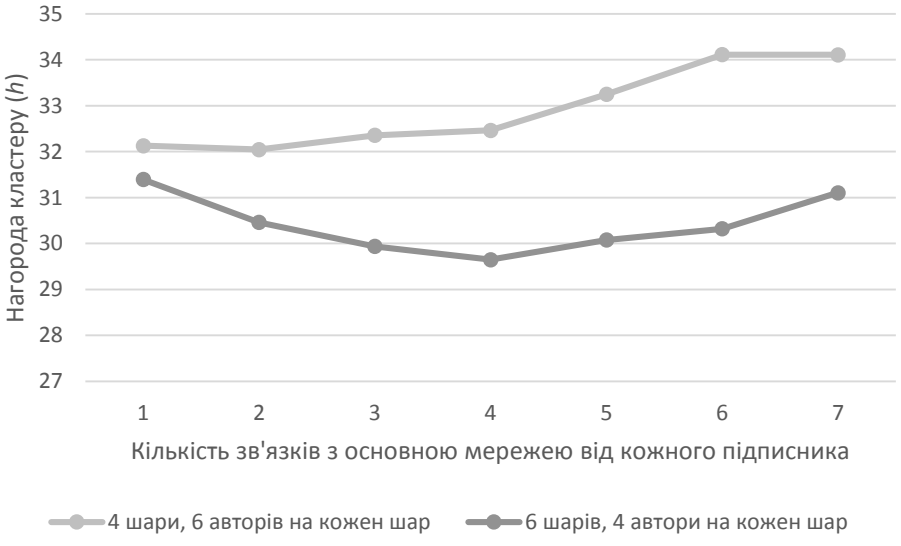


Рис. 1. Порівняння кластерів при використанні PA мережі і параметрів розподілу нагороди $\alpha_1=0.1, \alpha_2=0.8$

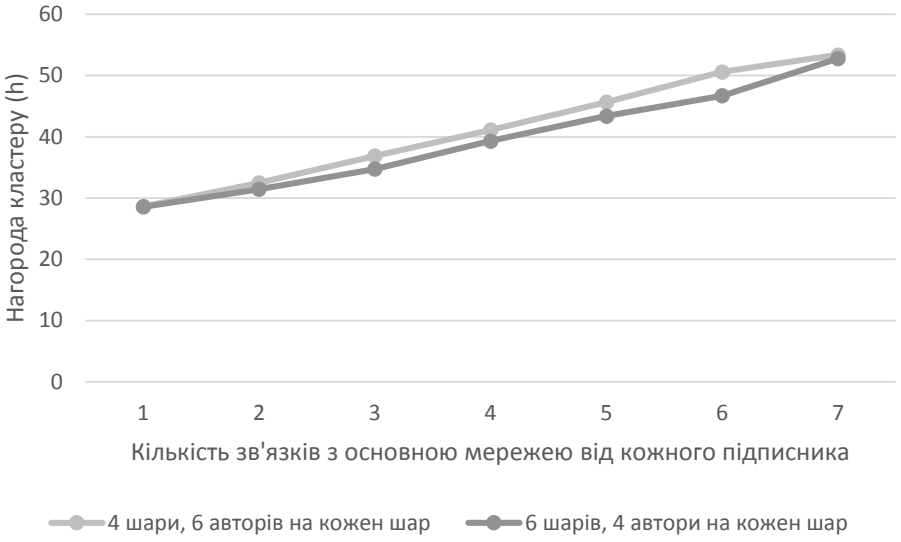


Рис. 2. Порівняння кластерів при використанні PA мережі і параметрів розподілу нагороди $\boldsymbol{\alpha_1 = 0.1, \alpha_2 = 0.5}$

Висновки

В даній роботі було запропоновано нову структуру для об’єднання спільноти free-ride користувачів і проаналізовано залежність кількості отримуваної цією спільнотою нагороди в залежності від кількості зв’язків від кожного з підписників спільноти free-ride користувачів до авторів, що не належать даній спільноті при використанні різної кількості шарів в запропонованій структурі і різної кількості авторів, на кожному з цих шарів. В подальшому планується дослідити способи модифікації алгоритму розподілу нагороди з метою обмеження можливостей зловживання за рахунок побудови великої кількості зв’язків, а також дослідити алгоритми пошуку спільнот free-ride користувачів в мережі [5].

Список літератури

1. Анищенко К.М., Жданова Е.Г., Скорик В.А., Сперкач М.О. Исследование возможностей злоупотреблений в децентрализованных сетях с распределением награды // INNOVATIVE SOLUTIONS IN MODERN SCIENCE. – 2019. – №2. – с. 46–62.
2. Алгоритм поиска компонент связности в графе. [Електронний ресурс] // Режим доступу: http://e-maxx.ru/algo/connected_components
3. Power laws and preferential attachment. [Електронний ресурс] // Режим доступу: http://snap.stanford.edu/class/cs224w-2015/slides/04-powerlaws.pdf
4. The Google Pagerank Algorithm and How It Works [Електронний ресурс] // Режим доступу: https://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm
5. Методы выделения сообществ в социальных графах. [Електронний ресурс] // Режим доступу: http://www.machinelearning.ru/wiki/images/8/8a/Nikishin_coursework_community_detection.pdf

УДК 004

Посыпайко Д. В.

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА ODOO

Интегрированное управленческое программное обеспечение сегодня является одним из ключевых источников конкурентного преимущества. Odoo - система ERP и CRM написана на языке программирования Python, выступающая инструментом, используемый для внедрения передовых систем управления в различных сферах деятельности, включая сельское хозяйство, открытые биржевые торги, торговые ассоциации и многие другие сферы.

Integrated management software today is one of the key sources of competitive advantage. Odoo is an ERP and CRM system written in the Python programming language, a tool used to implement advanced management systems in various fields of activity, including agriculture, open stock trading, trade associations and many other areas.

1. Введение

Интегрированное управленческое программное обеспечение сегодня является одним из ключевых источников конкурентного преимущества. Программное обеспечение Odoo состоит модулей такие как: бухгалтерский учет**,** CRM, управление персоналом, управление производством, продажами, закупками, складом, проектами, управление транспортом, претензиями, POS. Каждый модуль включает отдельный функционал. Наличие такого программного обеспечения в компании открывает следующие положительные аспекты для нее: повышение конкурентоспособности, уменьшение рисков деятельности, увеличение доходов, снижение издержек, автоматизация и анализ статистических данных и другие преимущества для бизнеса.

2. Структура сервиса

Фреймворк Odoo был выбран основываясь на его преимуществах. Поскольку такое программное обеспечение с открытым кодом дает новый способ решения проблем на современном этапе развития бизнеса. Модуль Odoo может содержать следующие элементы:

-Бизнес-объекты: объявленные как Python классы, эти ресурсы автоматически сохраняются в Odoo в соответствии с ее конфигурацией.

-Данные: XML или CSV файлы объявляют метаданные (представления или бизнес-процессы), данные конфигурации (параметризация модулей), демонстрацию данных и многое другое.

-Веб-контроллеры: обрабатывать запросы от веб-браузеров.

-Статические ресурсы: изображения, CSS или javascript-файлы, используемые веб-интерфейсом или веб-сайтом

Для создания WEB-приложения был использован высокоуровневый язык программирования Python и интегрированная среда разработки PyCharm. Odoo распространяются с помощью сервиса Google Play и App Store, что позволяет сделать его общедоступным для скачивания.

3. Odoo

Odoo — ERP и CRM-система, разрабатываемая бельгийской компанией Odoo S. A.

CRM (Customer Relationship Management), или «**управление отношениями с клиентами**». Программа которая помогает хранить и систематизировать данные о клиентах, заявках и сделках. Информация собрана в удобных карточках: имена, контакты, покупки, договоры, счета и платежи. Здесь же в хронологическом порядке хранится вся история работы с заказчиком, письма и записи звонков. Кроме того, система автоматизирует процессы и помогает менеджеру на каждом этапе продажи: напоминает позвонить клиенту, формирует документы по шаблону, выставляет счета, создает аналитические отчеты, отправляет sms, ставит задачи и контролирует их выполнение.

ERP (Enterprise Resource Planning), или **«планирование ресурсов предприятия».**

Программа хранит, обрабатывает и ведет единую базу данных компании, а также синхронизирует деятельность всех подразделений: отдел заказов, производственные цеха, склад, логистический отдел, бухгалтерию, отдел рекламы и т.д. ERP создает единое информационное пространство для всех сотрудников компании. Данные вносятся в сервис один раз, и становятся доступны для всех.

В стандартном “комплекте” Odoo включает в себя следующие приложения:

-Управление продажами (CRM, Sales, Invoicing, Point of Sale)

-CMS система (Web site builder, Интернет-магазин, Q&A Forum, Blogs, Slides, Live Chat)

-Операционная деятельность (Manufacturing, Purchase, Inventory, Human Resources, MRP, HelpDesk, Recruitment, Employees, Expenses, Appraisal, etc.) и др.

4. Основное преимущество

Ключевой компонент Odoo это ORM, который избавляет от необходимости писать самому SQL запросы руками и предоставляет гибкие и безопасные сервисы такие как API на разных языках для доступа к системам Odoo без явного обращения к XML-RPC или JSON-RPC.

Бизнес-объекты объявляются как расширяемые классы Python model, которая автоматически интегрирует их в систему хранения данных. Модели данных могут быть сконфигурированы настройкой некоторого количества атрибутов в их описании.

5. Разработка

В результате разработки было реализовано различные уровни доступа пользователей такие как директор, пользователь, системный администратор. Был реализован интерфейс для работы с CRM-системой (рис. 1), модификация существующей модели для оформления заказа (рис. 2), интеграция базы данных, а также функционал который отвечает за формирование и распространение ордеров и счетов (рис. 2).

Кроме этого для пользователей было реализовано добавление товара в каталог, обработка заказа, просмотр статистики, оформление покупки, начисление различных бонусов клиентам и демонстрация этой информации в чеке, возможность отследить всю историю продаж клиента и не только, а также возможность присвоения ролей пользователям для администратора.

Последнее позволяет администратору легко внедрять новых сотрудников в систему.

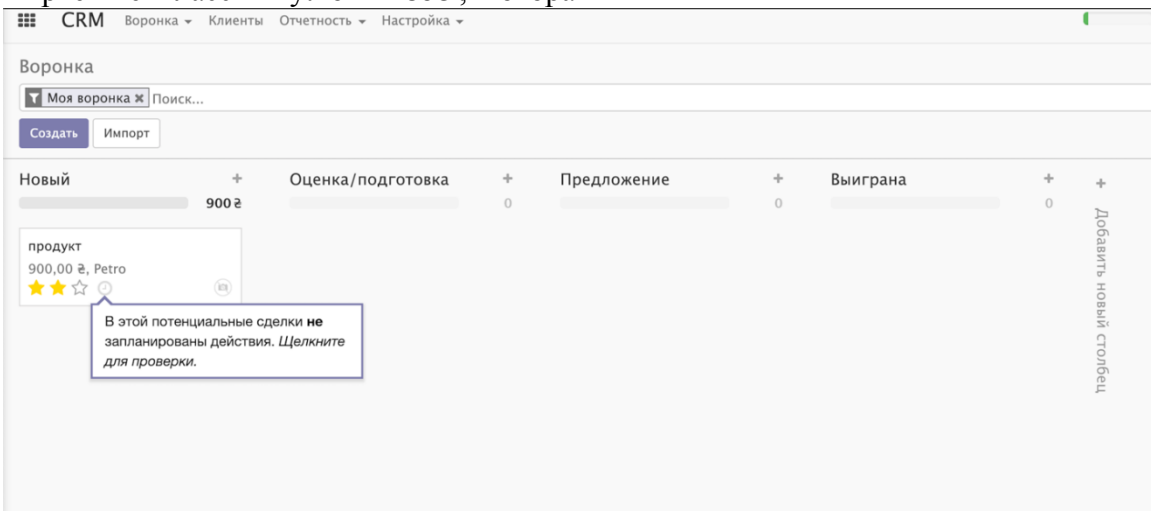


Рис.1. Интерфейс для работы с CRM

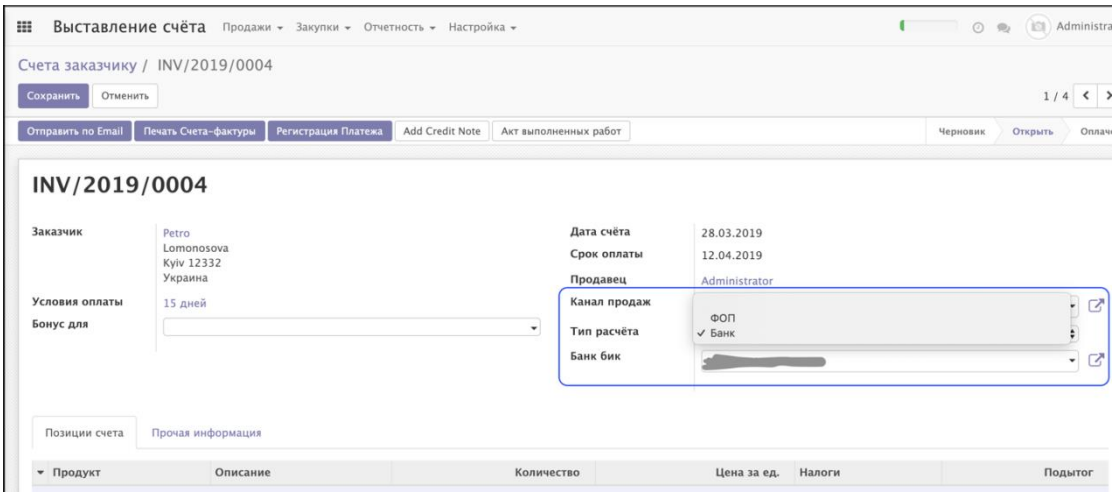


Рис.**2**. Модификация существующей модели для оформления заказа.

6. Заключение

В работе рассмотрена общая информация о программном продукте Odoo который использован при разработке WEB-приложения. Данный программный продукт выступает, на мой взгляд, одним из лучших инструментов, который оптимально подходит для внедрения передовых систем управления в различных сферах деятельности.

Список литературы

1. Odoo documentation - https://www.odoo.com/documentation/user/12.0/
2. ERP and SRM system - https://salesap.ru/otlichiya-srm-erp/

УДК 681.3.01

БЛАЖКО І.О.

АЛГОРИТМ ЗІСТАВЛЕННЯ ШАБЛОНІВ ПОШУКУ З ВХІДНИМИ РЯДКАМИ ДЛЯ ВИЗНАЧЕННЯ МОЖЛИВОСТЕЙ БРАУЗЕРА

В статті розглянуто алгоритм зіставлення шаблонів пошуку з вхідними рядками для визначення можливостей браузера, де вхідними рядками є заголовки запитів користувача user-agent, а шаблони пошуку представлені рядками зі спеціальними символами підстановки. Розглянуто та проаналізовано напрацювання в області вирішення задачі зіставлення шаблонів пошуку з вхідними рядками. Запропоновано новий алгоритм вирішення задачі, наведені кроки алгоритму та проаналізовано часову складність алгоритму для поставленої задачі. Зроблені порівняння швидкодії роботи отриманої програми з аналогами та продемонстровано перевагу запропонованого підходу.

КЛЮЧОВІ СЛОВА: ШАБЛОНИ ПОШУКУ, СИМВОЛИ ПІДСТАНОВКИ, МОЖЛИВОСТІ БРАУЗЕРА, АГЕНТ КОРИСТУВАЧА

The multi-pattern matching with wildcards problem is considered in this article for determining browser capabilities, given online requests that are user-agent strings and given predefined set of patterns that are represented as strings with wildcard characters. Related works and algorithms for multi-pattern matching were studied and the new algorithm was developed that solves given problem.

Steps of the algorithm were described, and the time complexity of the algorithm was analyzed. The program implementation was compared with most prominent analogues and the advantage of the proposed approach has been shown in terms of execution speed.

KEYWORDS: MULTI-PATTERN MATHCING, WILDCARDS, VARIABLE LENGTH DON'T CARE, BROWSER CAPABILITIES, USER-AGENT

1. Вступ

На сьогоднішній день для багатьох компаній та бізнесів які здійснюють свою діяльність в мережі інтернет необхідність у визначенні усієї можливої інформації про користувача стоїть дуже гостро. Один з найбільш дієвих шляхів отримання такої інформації - це аналіз user-agent заголовку запиту користувача, тому що використовуючи цей заголовок можна отримати інформацію про браузер користувача, платформу яку цей браузер використовує, перелік можливостей браузера та інше. Використовуючи отриману інформацію бізнес може налаштувати обробку запитів клієнтів в залежності від типу трафіку, чи то мобільний телефон чи то комп'ютер, можна вирішити який вміст буде відправлено до запитуючого пристрою, або навіть адаптувати вміст на льоту. Це особливо корисно, коли ви маєте справу з широким спектром пристроїв, що використовуються сьогодні, і дозволяє якнайкраще використовувати стратегію націлювання за змістом. Окрім веб-оптимізації, це має очевидне застосування в рекламному секторі, а саме у випадках коли пристрій користувача може виступати як критерій для визначення приналежності до цільового ринку. Не менш важливий варіант використання - це аналітика, завдяки наявності якомога повної інформації про клієнтів, подальший аналіз може суттєво покращити рішення щодо змісту контенту який публікується, стратегії націлювання або оптимізації процесів перетворення користувачів сайту на клієнтів. Наявність постійно оновлюваного методу аналізу user-agent заголовків також означає, що бізнес буде знати про появу нових пристроїв за допомогою яких відвідують їх ресурс, а отже пов'язані з цим проблеми будуть виявлені на ранній стадії.

Але аналіз user-agent заголовку не є тривіальним завданням. Існує багато бібліотек які вирішують цю задачу балансуючи між точністю та швидкістю отримання результатів. Існують навіть сервіси, що надають послуги по визначенню та аналізу user-agent заголовку, такі як deviceatlas.com. Проблема точності може бути вирішена використовуючи відкриті списки user-agent шаблонів, які постійно оновлюються, але наразі таких шаблонів більше ніж 200 тисяч[1] та наявні рішення невзмозі швидко з ними всіма працювати, в той час як швидкість повернення відповіді до клієнта є іншим життєво важливим показником для бізнеса. Тому виникає необхідність в швидких методах аналізу цих шаблонів, а саме методах зіставлення великої кількості шаблонів пошуку з user-agent екземплярами.

2. Змістовна постановка задачі

На вхід подається множина шаблонів пошуку:

$$P = \{p_1, p_2, \dots, p_m\}$$

Кожен шаблон пошуку p_i має вигляд рядку символів які належать алфавіту Σ, та спеціальних символів які не належать цьому алфавіт, ці символи ще називають символами підстановки:

«?» − відповідає будь-якому рівно одному символу з алфавіту Σ;

«*» − відповідає будь-якому рядку символів з алфавіту Σ, включаючи порожній рядок.

Тобто p_i можна представити як:

$$p_i = c_{i1}k_{i1}c_{i2}k_{i2}c_{i3} \dots c_{il_i}k_{il_i} *,$$

де c_{ij} є символом підстановки, а k_{ij} є ключовим словом – рядком символів алфавіту Σ. Для c_{i1} існує додаткова умова – він може бути або пропущений, або рівний «*».

Далі на вхід надходять запити наступного вигляду: рядок S, що складається з n символів. Треба для кожного такого рядка знайти усі шаблони пошуку з P, що відповідають цьому рядку, тобто отримати множину:

$$A = \{p_{a_1}, p_{a_2}, \dots, p_{a_{anslen}}\}$$

Кількість запитів не обмежена, потрібно відповідати на запити по мірі надходження.

До особливостей цієї задачі відноситься те, що множина шаблонів пошуку є дуже

великою – мільйони екземплярів. В той же час розміри рядків запитів зазвичай знаходяться в межах від 100 до 200 символів.

3. Огляд методів вирішення задачі зіставлення шаблонів пошуку з вхідними рядками

Перед початком розробки алгоритму зіставлення шаблонів пошуку з вхідними рядками на предмет збігу було зроблено огляд вже існуючих напрацювань у цій області.

На початку, було виконано огляд літератури з зіставлення шаблонів пошуку з рядками символів, а саме методи розбору регулярних виразів. Було виявлено, що більшість методів зосереджені на роботі з парами регулярний вираз та вхідний рядок. Такі методи добре працюють з парами, але зі збільшенням кількості регулярних виразів – їх швидкодія росте лінійно від кількості виразів. Також було зроблено розбір найбільш популярних алгоритмів для роботи з регулярними виразами, але як виявилось - вони потребують попередньої обробки регулярного виразу з дуже великими витратами пам’яті та часу.

Регулярні вирази є дорогими з точки зору обчислення. Тому їх доцільно використовувати лише тоді, коли це дійсно необхідно. Тим не менш, шаблони пошуку, а тобто рядки з символами підстановки, мають майже таку ж широку область використання як і регулярні вирази, але значно простіші в використанні та для них існують більш швидкі алгоритми обчислення.

Переглядаючи попередні дослідження, можна виявити, що існує досить мало праць пов’язаних з дослідженням саме зіставлення великої кількості шаблонів пошуку з текстом.

Проблема досить легко вирішується якщо взяти будь-який алгоритм зіставлення одного шаблону пошуку з одним екземпляром тексту та окремо зіставити усі наявні шаблони пошуку з заданим текстом. Але зрозуміло, що це не є ефективним алгоритмом для вирішення даної задачі, оскільки ми ніяк не використовуємо інформацію про текст яку ми отримаємо при кожному окремому порівнянні шаблону пошуку та тексту, а також ми ніяк не використовуємо інформацію про “схожість” шаблонів пошуку.

У 1997 Gregory Kucherov та Michael Rusinowitch[2] запропонували алгоритм для вирішення задачі зіставлення шаблонів пошуку з текстом з часовою оцінкою складності $O((|P| + |t|)log(|P|))$. У 2010 Zhang та інші[3] запропоновали пришвидшення цього алгоритму шляхом побудови скінченного автомата за алгоритмом Ахо-Корасік[4] та динамічному маркуванні вузлів «предків» – часова складність склала $O((|P| + |t|)\frac{log(k)}{\log log(k)})$. Обидва алгоритми зосереджені на роботі з одним екземпляром тексту та протягом роботи виконують витратні операції побудови та зміни автоматів, тому вони не є придатними для використання з багатьма екземплярами тексту. Також ці алгоритми працюють тільки з символами підстановки виду «*».

У 2011 році було представлено три алгоритми[5] зіставлення тексту з декількома шаблонами пошуку з гарними показниками часу роботи – рішення базувалось на швидкому перетворенні Фур'є (FFT). Перший з алгоритмів розроблений для невеликої кількості шаблонів пошуку, другий працює за допомогою кодування шаблонів та тексту в прості числа, заснованих на простому кодуванні набору шаблонів і тексту, а третій заснований на використанні відстані Хеммінга між бітовими векторами та підходить тільки для випадків коли кількість символів підстановки у шаблонах дуже мала. Всі ці 3 алгоритми мають ще один суттєвий недолік крім перерахованих вище – кількість символів підстановки має бути фіксованою.

Пізніше був представлений алгоритм[6] який працює з будь-якими діапазонами символів підстановки та час виконання якого залежить від кількості часткових збігів з шаблонами пошуку. При роботі використовується автомат побудований за алгоритмом Ахо-Корасік, вхідний текст сканується зліва направо та одночасно будуються усі можливі шаблони. На початку роботи усі «початки» шаблонів додаються до черги, тому оцінка знизу цього алгоритму складає щонайменше $\Omega(m)$, де m – кількість шаблонів пошуку.

У 2018 було запропоновано інший алгоритм[7], який працює з символами підстановки які задають діапазон кількості символів, алгоритм заснований на використанні суфіксного дерева. За допомогою цього підходу не можна задати

символ підстановки який би позначав рядок довільної довжини. Оцінка часової складності цього алгоритму $O(n + m\frac{n}{|\Sigma|}\sum_{i=0}^{l-1} G_i)$, де n - довжина вхідного тексту, m - кількість шаблонів пошуку, $|\Sigma|$ - кількість символів в алфавіті, G_i - середня довжина і-го діапазону в шаблоні пошуку. Тобто як бачимо, часова складність залишається досить великою та дуже залежить від заданих діапазонів символів підстановки.

4. Алгоритм зіставлення шаблонів пошуку з вхідними рядками

Алгоритм базується на використанні такої структури даних як префіксне дерево[8], куди будуть записані усі наявні шаблони пошуку. Хід побудови цього дерева нестандартний та нагадує побудову "стислого" префіксного дерева, тому що кожен шаблон буде представлений у вигляді рядку з ключових слів та спеціальними символами «*» та «?» між ними, тобто кожне ребро в цьому дереві містить інформацію про ключове слово переходу та спеціальний символ що йому передує. Дерево будується на основі вхідних шаблонів та при подальшій роботі алгоритму є незмінним. Для зменшення витрат пам'яті, а також простоти представлення – кожному ключовому слову серед множини усіх ключових слів шаблонів з P буде наданий деякий символ з алфавіту Δ, тобто існує бієктивне відображення таке що:

$$f: \delta_j \to w_j \text{ та } g: w_j \to \delta_j.$$

Для швидкого пошуку ключових слів у тексті запитів що надходять буде використовуватись автомат побудований за алгоритмом Ахо-Корасік[4]. Автомат буде побудований заздалегідь і лише раз на основі ключових слів з W. Це дозволить знаходити усі ключові слова в тексті S за час лінійний до кількості ключових слів в тексті. Далі при надходженні нового рядка буде проводитись його сканування зліва направо та ми одночасно будуть будуватись можливі шаблони пошуку використовуючи інформацію про переходи в префіксному дереві.

Алгоритм можна поділити на дві основні частини – попередня обробка шаблонів пошуку та інтерактивна частина відповіді на запити, тобто вхідні рядки. Розглянемо кроки запропонованого алгоритму:

Крок 1. Побудуємо набір усіх унікальних ключових слів W. Для кожного шаблону пошуку p_i додаємо k_{ij} до множини W:

$$W = \{w_j : j \in J\} = \bigcup_{i=1}^{m} \{k_{ij} : 1 \le j \le l_i\}$$

Крок 2. Кожному w_j з W поставимо у відповідність деяке число δ_j таким чином, щоб отримати бієкційне відображення:

$$f: \delta_j \to w_j \text{ та } g: w_j \to \delta_j.$$

Крок 3. Побудуємо множину B – перетворених шаблонів пошуку, шляхом заміни ключових слів в p_i на символи алфавіту Δ, при чому $|\Delta| = |W|$, та видаленню c_{i1} у разі його присутності:

$$B = \{b_i = g(k_{i1})c_{i2} \dots g(k_{il_i}) : 1 \le i \le m\}.$$

Крок 4. На основі B побудуємо префіксне дерево T наступним чином: кожен $g(k_{ij})$ будемо розглядати як звичайний символ алфавіту Δ, якщо c_{ij} дорівнює «*» або відсутній, та будемо додавати ребро з символом $g(k_{ij})$ до цього префіксного дерева. Оскільки розмір алфавіту може бути досить великим, то у кожному з вузлів будемо зберігати червоно-чорне дерево зі списком ребер – символів $g(k_{ij})$ в якості ключа для пошуку. Це дозволить швидко робити переходи по конкретним ребрам – символам алфавіту $|\Delta|$. Якщо c_{ij} дорівнює «?», то будемо додатково помічати такі ребра, щоб відрізняти їх від «звичайних» ребер.

Крок 5. Побудуємо скінченний автомат за алгоритмом Ахо-Корасік на основі ключових слів з W. Для кожного стану, що відповідає деякому ключову слову w_j будемо додатково зберігати δ_j. Переходи для кожного з станів по символам алфавіту Σ будуть зберігатись у червоно-чорному дереві.

Крок 5.1. Для всіх станів автомата з яких ми можемо потрапити в стан який відповідає деякому ключовому слову з W рухаючись по суфіксним посиланням ми попередньо обчислимо «кінцеві» $termLink$ посилання до найближчого з таких станів.

Крок 5.2. Для станів автомату що відповідають ключовому слову w_j, такому що присутнє у будь-якому шаблоні пошуку на першій позиції та при цьому не має символу підстановки що йому передує, ми додатково будемо зберігати посилання $headLink$ на

відповідний вузел у префіксному дереві T у який можна потрапити рухаючись від кореня T по $g(w_j)$.

Крок 5.3. Для станів автомату що відповідають ключовому слову w_j, такому що присутнє у будь-якому шаблоні пошуку на першій позиції та має символ підстановки «*» що йому передує, ми додатково будемо зберігати посилання $looseHeadLink$ на відповідний вузол у префіксному дереві T у який можна потрапити рухаючись від кореня T по $g(w_j)$.

Після виконання попередньої обробки шаблонів пошуку (перших п'яти кроків алгоритму), алгоритм може приймати запити аналізу текстів – вхідних рядків S. Для кожного окремого рядка S треба виконати наступні дії:

Крок 6. Створимо порожню масив q в якій будуть зберігатись пари значень: індекс рядка S та посилання на вузол префіксного дерева T. На початку роботи стан автомата відповідає кореневому вузлу $curState = rootState$.

Крок 7. Скануємо рядок S зліва направо та для чергового символу s_{idx} , де $1 \le idx \le n$ виконуємо наступні кроки:

Крок 7.1. Перейдемо з поточного стану автомата в новий за символом s_{idx} використовуючи функцію переходу в алгоритмі Ахо-Корасік та оновимо $curState$.

Крок 7.2. Якщо поточний стан відповідає деякому ключовому слову w_j та $|w_j| = idx$, тоді додамо пару $(idx, curState.headLink)$ до q.

Крок 7.3. Якщо поточний стан має посилання на $looseHeadLink$, тоді додамо пару $(idx, curState.looseHeadLink)$ до q.

Крок 7.4. Якщо поточний стан відповідає деякому ключовому слову w_j та $|w_j| \neq idx$, тоді $lb = idx - |w_j|$. Для всіх пар з q таких, що $q_i.idx \le lb$ перевірити чи є перехід з вузла дерева T по $g(w_j)$:

Крок 7.4.1. Якщо перехід існує та ребро не помічене символом «?», то виконаємо перехід по ньому, додавши пару idx та вузол до q.

Крок 7.4.2. Якщо перехід існує та ребро помічене символом «?», додатково перевіримо що $q_i.idx = lb - 1$ та додамо пару idx та вузол(на який перейшли) до q.

Крок 7.4.3. Якщо була додана нова пара значень до q, перевіримо чи відповідає вузол дерева T в цій парі якомусь шаблону пошуку, якщо так – то додамо цей шаблон пошуку до відповіді A.

Крок 7.5. Якщо поточний стан має посилання $termLink$, то робимо перехід за цим посиланням та повертаємось до кроку 7.3.

Крок 8. Повернути набір знайдених шаблонів пошуку A та закінчити роботу алгоритму.

Як бачимо в процесі роботи алгоритму ми рухаємось всіма можливими шляхами в префіксному дереві T, а це гарантує знаходження всіх шаблонів пошуку в рядку S, тому що вони всі є в T.

5. Аналіз часу виконання алгоритму та витрат пам'яті

Оцінку складності алгоритму можна поділити на дві частини: оцінка попередньої обробки шаблонів пошуку, яка виконується лише раз при ініціалізації програми та оцінка обробки чергового рядка S, що надходить до програми під час її роботи.

Розглянемо спочатку етап попередньої обробки шаблонів пошуку. Спочатку ми побудуємо множину всіх ключових слів за $O(|M|)$, де $|M|$ – сумарна довжина усіх шаблонів пошуку, нехай сумарна кількість ключових слів K. На кроці 3 ми витратимо $O(|M| + K\log|W|)$ на створення B. Побудова префіксного дерева займе $O(K\log|W|)$. Оцінка часу побудови скінченного автомата на кроці 5 є $O(|M|\log|\Sigma|)$. Тобто загалом, враховуючи залежності між змінними сумарна оцінка не перевищить $O(|M|\log|W|)$. При цьому витрати пам'яті є лінійними відносно сумарної довжини шаблонів пошуку, тобто мають оцінку $O(|M|)$.

Тепер перейдемо до оцінки часу та пам'яті обробки рядка S, що надходить під час роботи програми. Під час роботи алгоритму, ми проходимо через усі екземпляри знайдених ключових слів у рядку S, нехай їх кількість дорівнює occ. Для кожного знайденого ключового слова ми перевіряємо усі можливі стани в q з яких можливий перехід по цьому ключовому слову. Кількість таких станів є кількістю префіксів з T які були знайдені у частині рядка, що передує нашому ключовому слову. Тобто часову складність алгоритму можна оцінити як $O(occ * prefnum * \log|W| + anslen)$. Для гіршого випадку

$prefnum$ може бути оцінений як $O(|T|)$ у разі якщо ми не будемо рахувати дублікатів в алгоритмі як можливих різних відповідей, оскільки це не має сенсу для нас та оцінка може стати експоненційною в такому разі. Але наприклад оцінка знизу у випадку коли ми не маємо збігів з префіксами T становить $\Omega(occ)$. Тобто можна зробити висновок, що алгоритм мусить добре працювати для нашого типу даних, коли вхідні рядки S не перевантажені ключовими словами, а шаблони пошуку в своїй більшості не є підпослідовностями один одного.

6. Результати дослідження

Використовуючи алгоритм запропонований вище була розроблена програмна реалізація на Java. Програмна реалізація відрізняється від алгоритму деякими незначними деталями та способом зберігання даних, наприклад замість червоно-чорних дерев були використані хеш-таблиці, що працюють швидше на практиці для наших даних та операцій з ними.

У якості аналога для порівняння було обрано найкращу бібліотеку на даний момент[9] з точки швидкості аналізу user-agent рядків, яка використовує повний список шаблонів пошуку з бази даних browscap[1].

Для проведення бенчмаркінгу були використані інструменти JMH[10], ці інструменти є одними з кращих для проведення замірів часу виконання вашого Java коду в умовах наближених до реальних.

Час роботи був порівнянний для різних вхідних даних – реальних user-agent рядків. І було виявлено значну перевагу у швидкості обробки рядків розробленою програмою в порівнянні з бібліотекою browscap-java[9]. А саме, для рядка «Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36», який дуже гарно демонструє «середній» випадок ми отримаємо, що browscap-java бібліотека здатна на 6.978 ± 0.046 обробок запитів в мілісекунду, в той час як отримана програмна реалізація виконує 40.584 ± 0.415 обробок. Тобто ми отримали прискорення у 7 разів порівняно з бібліотекою аналогом. Також додатково ще розглядається можливість значного покращення отриманого результату шляхом оптимізації витратних операції для процесору та(чи) доступу до даних.

Висновок

Було сформульовано проблему недостатньої швидкодії наявних бібліотек для визначення можливостей браузера користувача(клієнта), які використовують базу даних browscap. Зроблено детальний аналіз літератури, наявних досліджень та напрацювань в області зіставлення шаблонів пошуку з текстом. Розроблений алгоритм зіставлення довільної кількості шаблонів пошуку з екземплярами текстів в режимі реального часу. Було проаналізовано часову складність алгоритму та витрати пам’яті. За допомогою програмної реалізації було показано значну перевагу у швидкодії в порівнянні з існуючими аналогами виявлення можливостей браузерів.

Подальші дослідження можуть бути зроблені у напрямку оптимізації витрат пам’яті під час роботи алгоритму, а також зменшення часу попередньої обробки шаблонів пошуку – ініціалізації бібліотеки. Також можна розширити можливості задання символів підстановки, а саме зробити можливим вказування діапазонів кількості символів що можуть бути підставлені – це може бути корисно для отримання більш чіткого контролю над заданням шаблонів пошуку, але наразі ця можливість не підтримується базою даних browscap.

Перелік посилань

1. Browser Capabilities Project [Електронний ресурс] – Режим доступу до ресурсу: https://browscap.org/.
2. KUCHEROV, Gregory; RUSINOWITCH, Michaël. Matching a set of strings with variable length don't cares. Theoretical Computer Science, 1997, 178.1-2: 129-154.
3. ZHANG, Meng; ZHANG, Yi; HU, Liang. A faster algorithm for matching a set of patterns with variable length don't cares. Information Processing Letters, 2010, 110.6: 216-220.
4. AHO, Alfred V.; CORASICK, Margaret J. Efficient string matching: an aid to bibliographic search. Communications of the ACM, 1975, 18.6: 333-340.

5. ZHANG, Meng, et al. Multi-pattern Matching with Wildcards. JSW, 2011, 6.12: 2391-2398.
6. HAAPASALO, Tuukka, et al. Online dictionary matching with variable-length gaps. In: International Symposium on Experimental Algorithms. Springer, Berlin, Heidelberg, 2011. p. 76-87.
7. Liu, N., Xie, F. & Wu, X. Pattern Anal Applic (2018) 21: 1151. https://doi.org/10.1007/s10044-018-0733-0
8. MEHTA, Dinesh P.; SAHNI, Sartaj. Handbook of data structures and applications. Chapman and Hall/CRC, 2004.
9. browscap-java [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/blueconic/browscap-java.
10. Java Microbenchmark Harness [Електронний ресурс] – Режим доступу до ресурсу: https://openjdk.java.net/projects/code-tools/jmh/.

УДК 519.85.004.42

Пономаренко И. Р.

РАЗРАБОТКА МИКРОСЕРВИСОВ НА ОСНОВЕ VERT.X

В настоящее время распространен термин «микросервисная архитектура» как способ разработки приложений в виде набора независимо развертываемых сервисов. Приложение состоит из набора небольших сервисов, каждый из которых работает в собственном процессе и взаимодействует с остальными с помощью легковесных механизмов, как правило HTTP. Используя совокупность разбитых на мелкие гранулы архитектур микросервисов, можно ускорить поставку программного обеспечения и внедрить в практику самые новые технологии.

Currently, the term “microservice architecture” is common as a way to develop applications as a set of independently deployable services. HTTP consists of many small services, each of which works in its own process and interacts with the rest with the help of lightweight images. With the help of architectural microservices scattered over small granules, it is possible to accelerate the delivery of software and put into practice the newest technologies.

1. Введение

В настоящее время распространен термин «микросервисная архитектура» как способ разработки приложений в виде набора независимо развертываемых сервисов. Приложение состоит из набора небольших сервисов, каждый из которых работает в собственном процессе и взаимодействует с остальными с помощью легковесных механизмов, как правило HTTP. Используя совокупность разбитых на мелкие гранулы архитектур микросервисов, можно ускорить поставку программного обеспечения и внедрить в практику самые новые технологии.

2. Структура сервиса

Такого рода сервисы построены на основе бизнес-потребностей и развертываются независимо с использованием полностью автоматизированной среды. Сами по себе эти сервисы могут быть написаны на разных языках и использовать разные технологии хранения данных.

Корпоративные приложения часто включают три основные части: пользовательский интерфейс (состоящий, как правило, из HTML страниц и javascript кода), база данных (как правило, реляционная) и сервер. Монолитный сервер — довольно очевидный способ построения подобных систем. Вся логика по обработке запросов выполняется в единственном процессе, само приложение разделено на классы, функции и namespaces.

Архитектура микросервисов использует библиотеки, но основной способ разбиения приложения — деление его на сервисы. Сервисы — это компоненты, выполняемые в отдельном процессе и

взаимодействующие между собой через веб-запросы или remote procedure call.

3. Eclipse Vert.x

Eclipse Vert.x – это проект с открытым исходным кодом от Eclipse Foundation, который предоставляет удобную модель для построения микросервисов. Vert.x имеет API для разработки асинхронных сетевых приложений, и можно подобрать необходимые модули для приложения: взаимодействия с БД, мониторинга, аутентификации, логгирования, обнаружения сервисов и т.д. Vert.x – платформа полиглот, поддерживает ряд языков, использующих JVM, возможна разработка частей приложения на различных JVM языках.

Основные понятия, которые использует Vert.x – это verticle (вертикаль) и event bus (шина событий), позволяющая вертикалям взаимодействовать между собой.

Вертикаль – единица развертывания в Vert.x, она обрабатывает входящие события, приходящие в event loop, такие как получение входящих сетевых буферов или сообщений от других вертикалей. Vert.x API неблокирующее, поэтому важные концепции Vert.x следующие [2]:

• каждое событие должно быть обработано за приемлемое время и не блокировать поток – event loop;

• блокирующие операции не должны выполняться в потоке, связанном с event loop.

Вертикаль всегда выполняется в одном потоке, и нет необходимости заботиться о проблемах многопоточности внутри вертикали. Vert.x использует все CPU (или ядра CPU) компьютера путем создания потока на CPU. Один поток может отправлять сообщения многим вертикалям.

Входящие сетевые данные, которые получает поток – event loop, передаются в качестве событий соответствующим вертикалям. Вертикаль может быть развернута 37 несколько раз, она может развернуть другую вертикаль. Если вертикаль развернута более чем один раз, то события распределяются между экземплярами вертикали в порядке круговой очереди. Обработка события заключается в выполнении соответствующего обработчика (listener object), зарегистрированного вертикалью. После этого поток может отправлять следующее сообщение другой вертикали.

4.Основное преимущество

Event bus – связующее звено Vert.x приложения, позволяющее различным частям приложения, независимо от того на каком языке они написаны, взаимодействовать друг с другом (рис. 1). Даже client-side JavaScript код, выполняющийся в браузере, может общаться на той же шине (event bus). В соответствующем API поддерживается сообщения типа издатель/подписчик, точка-точка, запрос-ответ.

Vert.x естественно интегрируется с RxJava – популярной библиотекой для асинхронной обработки потоков данных, основанной на паттерне проектирования Observer.

Vert.x построен как имплементация уже классического паттерна Reactor с маленькой модификацией, которую разработчики прозвали Multi-Reactor.

5. Паттерн Reactor

Чтобы понять паттерн Multi-Reactor, достаточно знать известный паттерн Reactor. Классический Reactor говорит о том, что есть некий Event Loop, как правило однопоточный, который отвечает за обработку событий. Все клиентские запросы заходят как события. Далее выполняется обработчик, Handler, который подписан на соответствующие события. При этом будет нехорошо, если обработчик заблокирует Event Loop надолго. Поэтому долгоиграющие задачи делегируются Worker-потокам и выполняются, не блокируя Event Loop. На них повешен некий Callback, который будет вызван, как только задача будет выполнена (или прервется с отчетом об ошибке).

В свою очередь, Multi-Reactor расширяет этот шаблон (паттерн), добавляя еще несколько потоков (дополнительные Event Loop-ы). Таким образом, формируется шина событий (Event Bus) которая умеет масштабироваться под ресурсы конкретной машины. Как правило, количество потоков Event Loop определяется по формуле «количество ядер процессора * 2». Итого, весь Vert.x — это один большой Event Bus, с которым мы общаемся посредством Callback-ов.

6.Разработка

Мы рассматриваю Vert.x как надежный инструмент для разработки системы, где важна высокая производительность. Он очень быстрый, потребляет мало ресурсов и очень стабилен, хоть и немного непривычен. Также нужно отметить риски, связанные с поддержкой и дальнейшим развитием этого инструмента. Все проекты, использующие Vert.x, которые были рассмотрены, оказались очень удачными (с технической точки зрения).

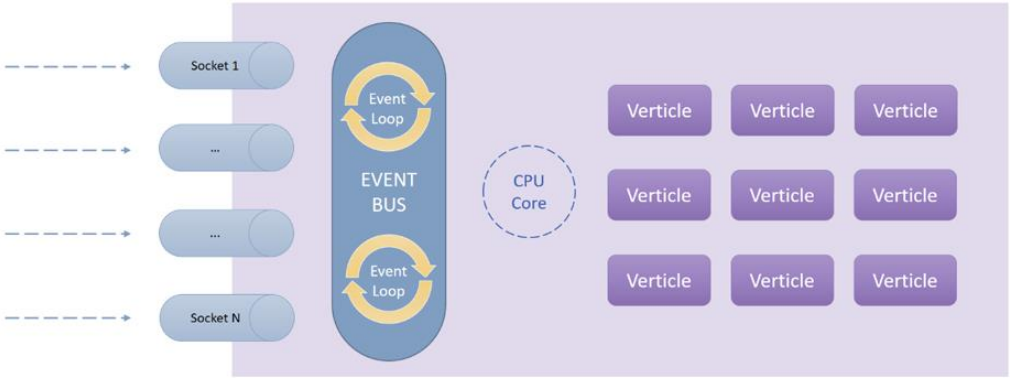


Рис.2.Структура Vert.x

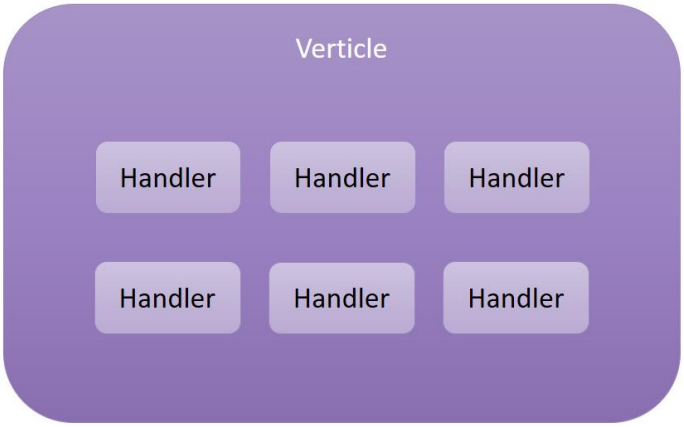


Рис.2.Структура Verticle

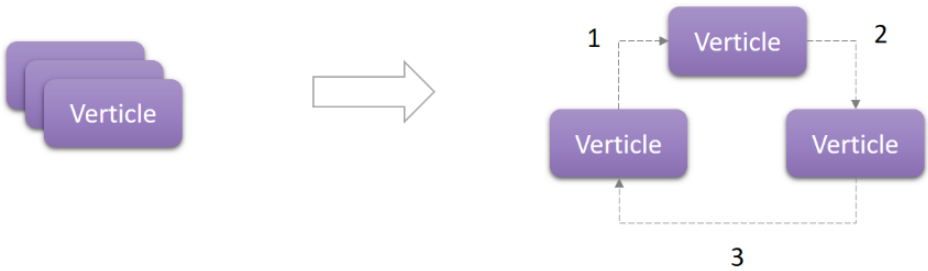


Рис.3.Балансировка нагрузки

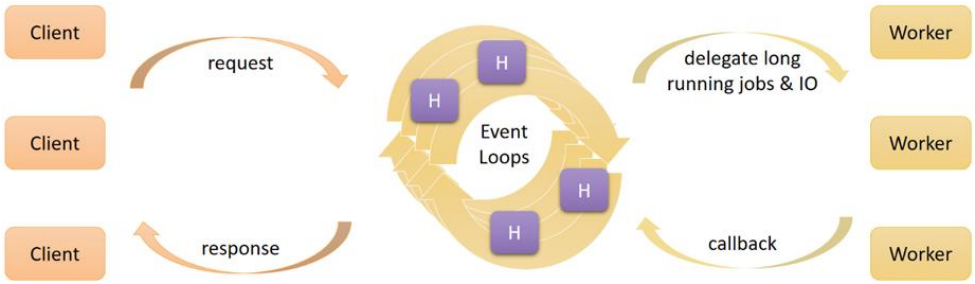


Рис.4. Паттерн Reactor

Заключение

Таким образом, в докладе рассмотрена общая информация о разработке микросервисного приложения с помощью платформы Vert.x, которая является асинхронной и неблокирующей. Следует отметить отличие ее многопоточной модели от модели, принятой во многих фреймворках, когда каждому сетевому клиенту назначается отдельных поток, что может нанести ущерб масштабируемости приложения, а также при большой нагрузке операционная система значительное время тратит на управление диспетчеризацией потоков.

Список литературы

1. Vert.x documentation. – Mode of access: http://vertx.io/docs/ .
2. A gentle guide to asynchronous programming with Eclipse Vert.x for Java developers. – Mode of access: http://vertx.io/docs/guide-for-java-devs/.

УДК 347.7

ЖИДКОВ В.О
ПРОСКУРА С.Л.

ІНФОРМАЦІЙНА СИСТЕМА ПІДТРИМКИ РОБОТИ МАГАЗИНУ ДИТЯЧИХ ТОВАРІВ

В даній статті розглянуто спрощення управління магазином дитячого одягу. Демонструється використання математичного алгоритму для прогнозування необхідних закупівель для магазину. Використання даної системи дозволить значно прискорити виконання необхідних бухгалтерських дій та значно спростить керування магазином.

КЛЮЧОВІ СЛОВА: ФОРМУВАННЯ ЗВІТНОСТІ, ПРОГНОЗУВАННЯ ЗАКУПІВЛІ, УПРАВЛІННЯ МАГАЗИНОМ

The subject of the article is an application of the simplification of the management of children's clothing store is considered. Demonstrates the use of a mathematical algorithm to predict the required purchases for a store. Using this system will significantly accelerate the implementation of necessary accounting actions and significantly simplify the management of the store.

KEYWORDS: REPORT FORMATION, PURCHASE FORECASTING, STORE MANAGEMENT

1. Вступ

На сьогоднішній день торгівля являється невід'ємною частиною нашого життя, все що нам необхідно ми отримуємо купівлею: їжа, одяг, товари для дому, і тд. Завдяки торгівлі люди проводять обмін речами, обмінюючи гроші на необхідні їм речі. У зв'язку з цим, торгівля розвивається дуже стрімко, можна здобути все більше різноманітних речей, які поліпшать життя.

Не дивлячись на стрімкий розвиток, у малих підприємствах облік товарів здійснюється елементарними записами у зошити чи блокноти. Для поліпшення їхньої роботи можна розробити програмне забезпечення, яке поліпшить їм облік свого товару, що дозволить зберегти багато часу та виконувати більший обсяг роботи за коротший час.

Суть даної роботи полягає у створенні мобільного програмного продукту, який дозволить керувати магазином якомога простіше, швидве та практичніше.

Дане застосування може бути використане у магазині дитячого одягу. З його допомогою, можна буде легко керувати обліком товару у магазині, отримувати необхідні звітності та спростить сам процес продажу товару.

У даної системи існують аналоги такі як «Товари, Ціни, Облік ...» та «Торгософт»[2], але вони мають свої недоліки, тому розробка даної системи є доцільна.

2. Постановка задачі

Управільння магазином завжди зводиться до таких ключових потреб:

- формування фінансової звітності;
- формування звітності по залишкам;
- формування звітності по проданому товару;
- формування звітності по списаному товару;
- продаж товару;
- списання товару;
- повернення товару;
- редагування каталогу;
- додавання товару у каталог;
- видалення з каталогу. [3]

Вирішення цих задач є достатньо тривіальними, і зводятся до елементарного відбору, групуванню та сортуванню необхідних даних для простішого пошуку та сприйняття необхідної інформації.

Складнішою задачею є прогнозування майбутніх закупок, оскільки необхідно враховувати об'єм продукції яка була продана і сам час, тобто необхідно розрахувати співвідношення проданого товару до часу і якщо це співвідношення буде доволі велике, то це буде означати, що на цей товар великий попит і необхідно закупити його ще.

Задачею прогнозування майбутніх закупок є порада або повідомлення для користувача, що на певний товар присутній великий попит. Також порада враховує й наявність цього товару у магазині та його кількість. [1] Дано:

Кількість залишку товару у магазині(n);

Кількість проданого товару(m);

Дати продажу кожної одиниці товару ($D[m]$);

Розміри товару (динамічна матриця K розмірністю r (кількість досліджуваних років) на q (4 сезони) на s (кількість розмірів товару)).

Відповідно, для коректного вирішення всіх вищеперерахованих задач, стає необхідним створити програмне забезпечення для спрощення керування магазином, зокрема магазином дитячого одягу.

3. Прогнозування майбутніх закупок.

Задача може бути вирішена наступним чином, залежно від теперішнього сезону необхідно знайти співвідношення кількості продукції, яка була загалом закуплена у кожному році, до кількості проданої продукції того ж року, а далі будуть надаватися прогнози у вигляді порад стосовно товарів, які було б бажано закупити або докупити для цього сезону, а також у кінці кожного сезону будуть надаватися поради для наступного сезону, також можна буде проглянути як у кожному році змінювалась тенденція на цей товар (переглянути дані стосовно кількості продаж за кожен рік та сезон).

Спершу ми маємо розподілити кожну одиницю проданого товару по рокам і сезонам, згрупувавши по розмірам (для розрахунків будемо використоваувати лише останні декілька років, оскільки попередні роки вже не актуальні). Для них знаходимо суму по кількості. Далі, якщо для товару кількість продажу кожен рік приблизно однакова або зростає, необхідно відзначити, що цей товар необхідно закупати. У випадку коли цей товар є на складі, необхідно вказати про необхідність докупити цей товар, якщо його недостатньо. У іншому випадку, вказати, що цей товар не потрібно закупати. Алгоритм проводится окремо для кожного товару.

Для більш детального пояснення наведений псевдокод алгоритму:

Крок 0. Для кожного товару.

Крок 1. Визначити s.

Крок 2. Згенерувати матрицю розмірністю 3x4xs та заповнити її нулями.

Крок 3. Для кожної одиниці товару.

Крок 3.1. ЯКЩО місяць продажу дорівнює 12, або 1, або 2 ТО *q = 1* ШНАКШЕ крок 3.2.

Крок 3.2. ЯКЩО місяць продажу дорівнює 3, або 4, або 5 ТО *q = 2* ШНАКШЕ крок 3.3.

Крок 3.3. ЯКЩО місяць продажу дорівнює 6, або 7, або 8 ТО *q = 3* ШНАКШЕ q = 4.

Крок 3.4. ЯКЩО рік продажу цей ТО *r = 1* ШНАКШЕ крок 3.5.

Крок 3.5. ЯКЩО рік продажу минулий ТО *r = 2* ШНАКШЕ крок 3.6.

Крок 3.6. ЯКЩО рік продажу позаминулий ТО *r = 3* ШНАКШЕ СТОП

Крок 3.7. K_{rqs} збільшуємо на 1.

Крок 4. Для кожного r

Крок 4.1. Для кожного q

Крок 4.1.1. ЯКЩО $K_{rq2} >= K_{rq3} - 0.05 * K_{rq2}$ і $K_{rq3} != 0$ і $K_{rq2} != 0$ ТО порадити закупити щонайменше $K_{rq2} - n$ ІНАКШЕ Крок 4.1.2.

Крок 4.1.2 ЯКЩО $K_{rq1} > 0$ і $n = 0$ і $K_{rq3} = 0$ і $K_{rq2} = 0$ ТО порадити докупити ще ІНАКШЕ порадити не закупати

Крок 5. Надати числові дані коритсувачу

4. Аналіз досліджень

Під час керування магазином його працівники проходять через певні процеси, поки не отримають прибуток від роботи магазину. Приблизний процес управління магазином наведено у діаграмі діяльності (рис. 1).

На початку роботи програми необхідно зчитати дані з бази даних, тому початок починається з підключення до бази даних та збереження їх у програмі для простішого оперування. А далі користувач може виконувати будь яку з наведених дій. У процесі роботи програми, усі дані зберігаються у самій програмі, тож кожна функція об’єкта StoreManage взаємодіє з DataBase. Order використовується лише для створення замовлення при продажі товару, тош викликається лише один раз.

5. Вхідні дані

Вхідні дані для системи обліку товару надаються користувачем у наступному виляді:

Додавання даних у каталог:

- найменування товару;
- код товару;
- розмір – розмір товару;
- кількість – кількість товару кожного розміру, який отрумується;
- можливість повернення;
- ціна – оптова ціна одиниці товару;
- відсоток націнки – відсоток, на який націниться товар;
- роздрібна ціна – роздрібна ціня одиниці товару.
- сезон у якому використовується;

Продаж товару, списання товару, повернення:

- найменування товару/код товару.

Перегляд звітів:

- період, який цікавить;
- найменування/код/сезон шуканого товару.

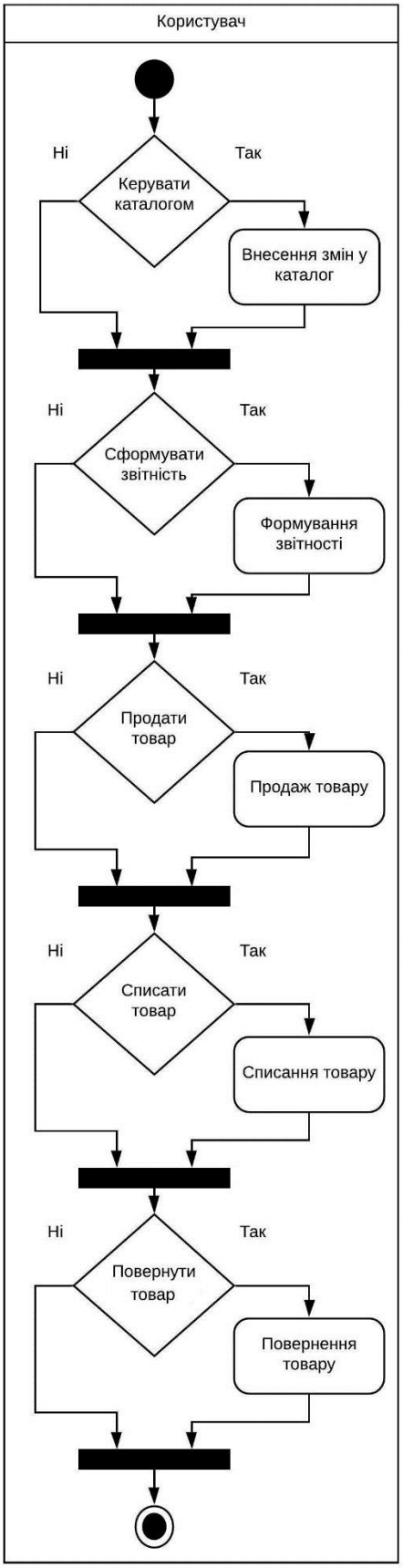


Рис. 1. Схема стурктурна діяльності. Процес управління магазином

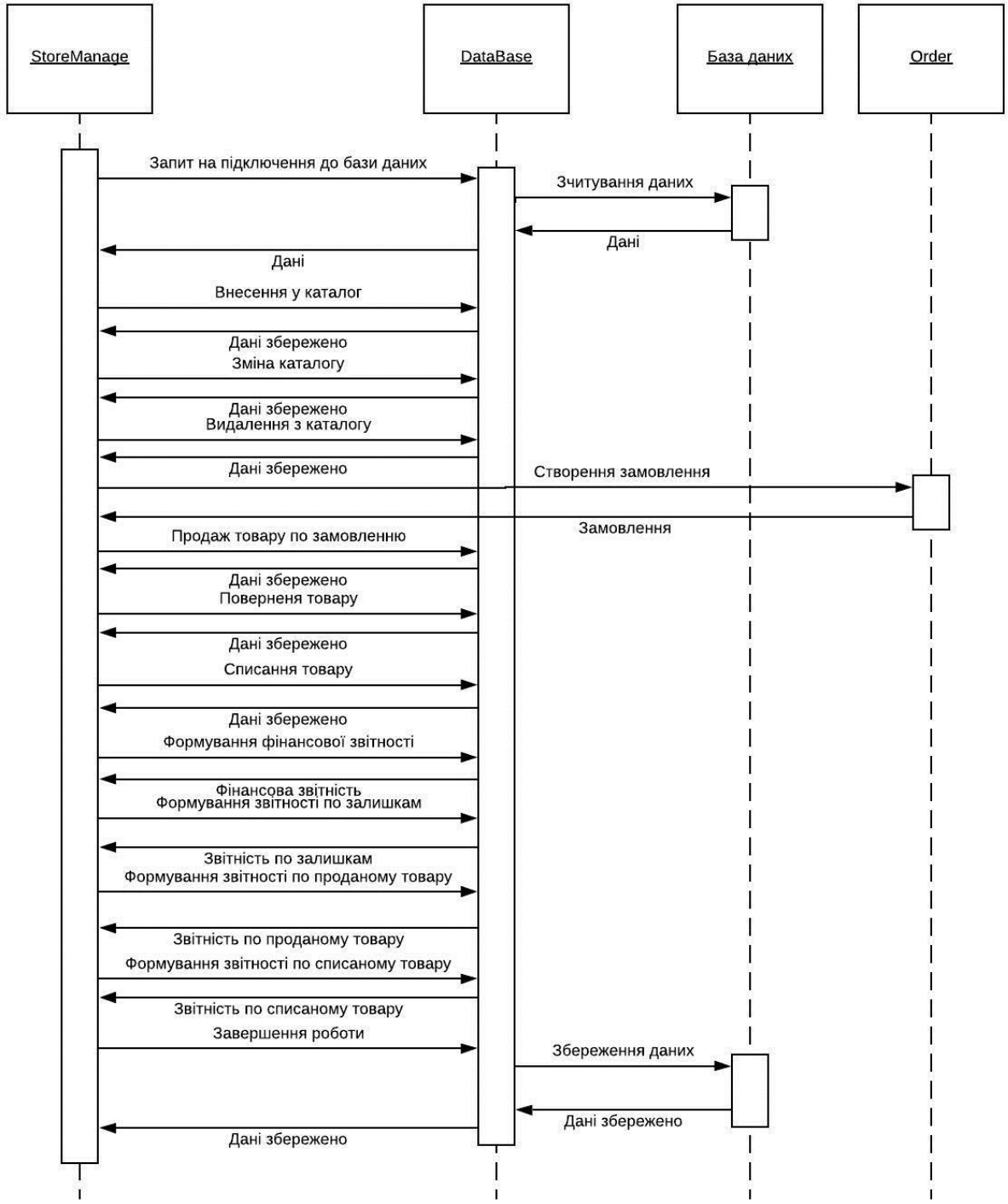


Рис. 2. Схема стурктурна діаграми послідовності

При написанні даного програмного продукту використовувалась реляційна база даних MySQL , інтегроване середовище Android Studio на платформі Android. База даних містить 4 таблиці:

– Товар – вона призначеня для зберігання інформації стосовно кожного товару;
– Розміри товару – призначена для зберігання всіх можливих розмірів товарів;
– Склад – призначена для поєднання таблиці Товар та Розміри товару;
– Продаж – призначення для зберігання інформації про товар, який було продано;
– Списання – призначення для зберігання інформації про товар, який було списано.

А також наявні такі зв'язки:

– Між таблицями Товар та Склад – зв’язок один до багатьох відповідно, оскільки для кожного товару може бути безліч розмірів з якими він поєднується у цій таблиці;
– Між таблицями Склад та Продаж – зв’язок один до багатьох відповідно, оскільки можна продати декілька одиниць товару одного й того ж розміру;
– Між таблицями Склад та Списання, оскільки можна списати декілька товарів одного й того ж розміру.

Табл. 1. Детальний опис бази даних

Назва таблиці	Назва стовпця	Тип даних	Призначення
Товар	Код	Decimal(100,0)	Ключове поле
	Найменування	Nvarcar(20)	Найменування товару
	Можливість повернення	Bool	Визначає чи можливо повернути товар у магазин чи ні
	Ціна оптова	Float(5,2)	Оптова ціна на товар
	Ціна роздрібна	Float(5,2)	Роздрібна ціна на товар
Склад	Код	Decimal(100,0)	Ключове поле
	Код товару	Decimal(100,0)	Зовнішній ключ
	Код розміру	Decimal(100,0)	Зовнішній ключ
	Кількість	Int(4)	Кількість товарів певного розміру
Розміри товару	Код	Decimal(100,0)	Ключове поле
	Розмір	Float(5,2)	Розмір товар
Продаж	Код	Decimal(100,0)	Ключове поле
	Код складу	Decimal(100,0)	Зовнішній ключ
	Кількість	Int(4)	Кількість одиниць проданого товару
	Дата продажу	Date	Дата продажу товару
Списання	Код	Decimal(100,0)	Ключове поле
	Код складу	Decimal(100,0)	Зовнішній ключ
	Кількість	Int(4)	Кількість одиниць списаного товару
	Дата списання	Date	Дата списання товару

6. Вихідні дані

На виході маємо форму, яка відображає дані, на які був запит від клієнта, а конкретніше один з варіантів:

- звіт по фінансам;
- звіт по залишкам;
- звіт по проданому товару;
- звіт по списаному товару.

Висновки

В даній статті було обґрунтовано доцільність створення системи спрощення управління магазином дитячого одягу. Був наведений математичний алгоритм прогнозування майбутніх закупок. Було описано необхідні дані та можливі дії системи.

Список літератури

1. Н.В. Дикань, І.І. Борисенко Менеджмент – 2012 – ст. 20-50
2. https://torgsoft.ua/ua/articles/upravlenie-magazinom/
3. http://catmanretail.com/biznes-treningi/uplravlenie-magazinom/
4. Лишиленко А.В. Бухгалтерський облік – 2017

УДК 004.9

ДОРОШЕНКО А.В.,
КОЧУБЕЙ І.Ю.,
ЖУРАКОВСЬКА О.С.

СИСТЕМА ДЛЯ СТВОРЕННЯ КАТАЛОГІВ ПОСЛУГ ТА ТОВАРІВ З ІНТЕГРАЦІЄЮ РЕКОМЕНДАЦІЙНОЇ ПІДСИТЕМИ

В даній статті освітлене питання створення каталогів послуг та товарів. Розглянуті існуючі системи створення веб-каталогів та описано реалізовану систему конфігурування каталогів послуг та товарів з використанням рекомендаційної підсистеми.

РЕКОМЕНДАЦІЙНА ПІДСИСТЕМА, КАТАЛОГ, ТОВАРИ, ПОСЛУГИ

The issues of creation catalogs of services and goods are highlighted in this article. The exist systems of creation web-catalogs are considered and implemented system of configuring catalogs of services and goods, which uses recommender subsystem, are described.

RECOMMENDER SUBSYSTEM, CATALOG, GOODS, SERVICES

Вступ

В час масового розповсюдження послуг та товарів через мережу Інтернет актуальними стали системи підтримки процесу створення та конфігурування каталогів вище згаданих послуг та товарів.

Важливою проблемою у сфері розповсюдження послуг та товарів є позиціювання пропозиції з врахуванням вподобань споживачів. Цю проблему допомагають вирішити алгоритми кластеризації та фільтрації даних.

Реалізована система поєднує в собі засоби створення веб-каталогів та алгоритми сегментації споживачів.

Постановка задачі

Цілями розробки системи для конфігурування каталогів послуг та товарів являються:

1. спрощення процесу кофігурування каталогів послуг та товарів;
2. покращення якості інформації, що пропонується споживачам за рахунок реалізації рекомендаційної підсистеми.

Для досягнення цілей необхідно вирішити наступні задачі:

1. спроектувати архітектуру системи;
2. створити схему бази данних;
3. створити інтерфейси для кофігурування каталогів;
4. реалізувати оплату послуг та товарів в рамках веб-каталогу;
5. оглянути існуючі рекомендаційні алгоритми;
6. розробити алгоритм надання рекомендацій.

Аналіз досліджень

Було розглянуто два рішення для швидкого конфігурування каталогів: «Wix.com» та «Joomla!», та зроблений порівняльний аналіз з реалізованою системою.

Базовий функціонал систем подібний. Всі розглянуті системи інкапсулюють в собі створення html-розмітки, sql-запитів, тощо.

Недоліком реалізованої системи є відсутність drag & drop редактора інтерфейсів.

Перевагою є можливість додати до функціоналу каталогу рекомендаційну підсистему.

Огляд реалізованої системи

Архітектура системи спроектована таким чином щоб розмежувати роботу з базою даних та реалізцією бізнес-логіки [1]. На рисунку 1 зображена діаграма розгортання.

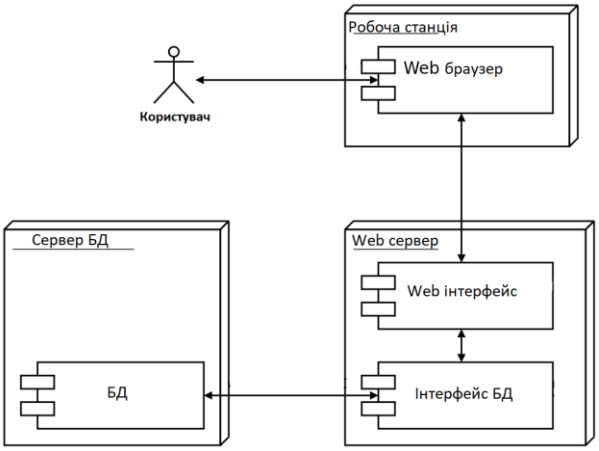


Рис. 1. Діаграма розгортання

Актори системи: розробник – суб'єкт, що створює каталоги послуг та товарів, адміністратор – суб'єкт, що наповнює контентом каталоги, користувач – суб'єкт, що має доступ до інформації про послуги та товари. На рисунку 2 зображена use-case діаграма.

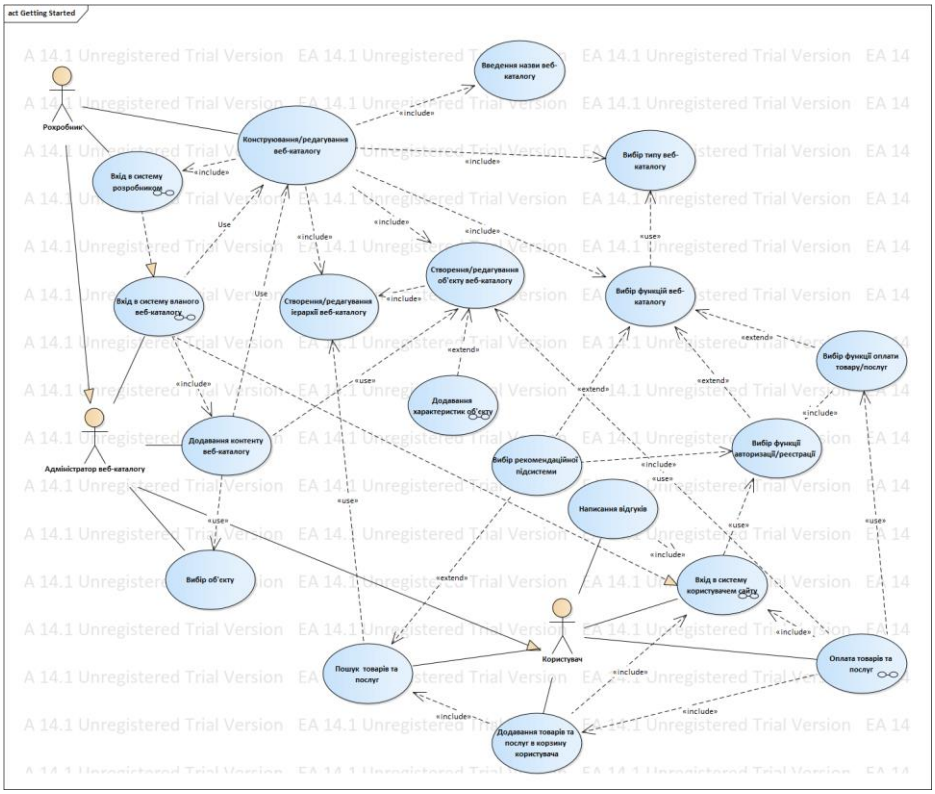
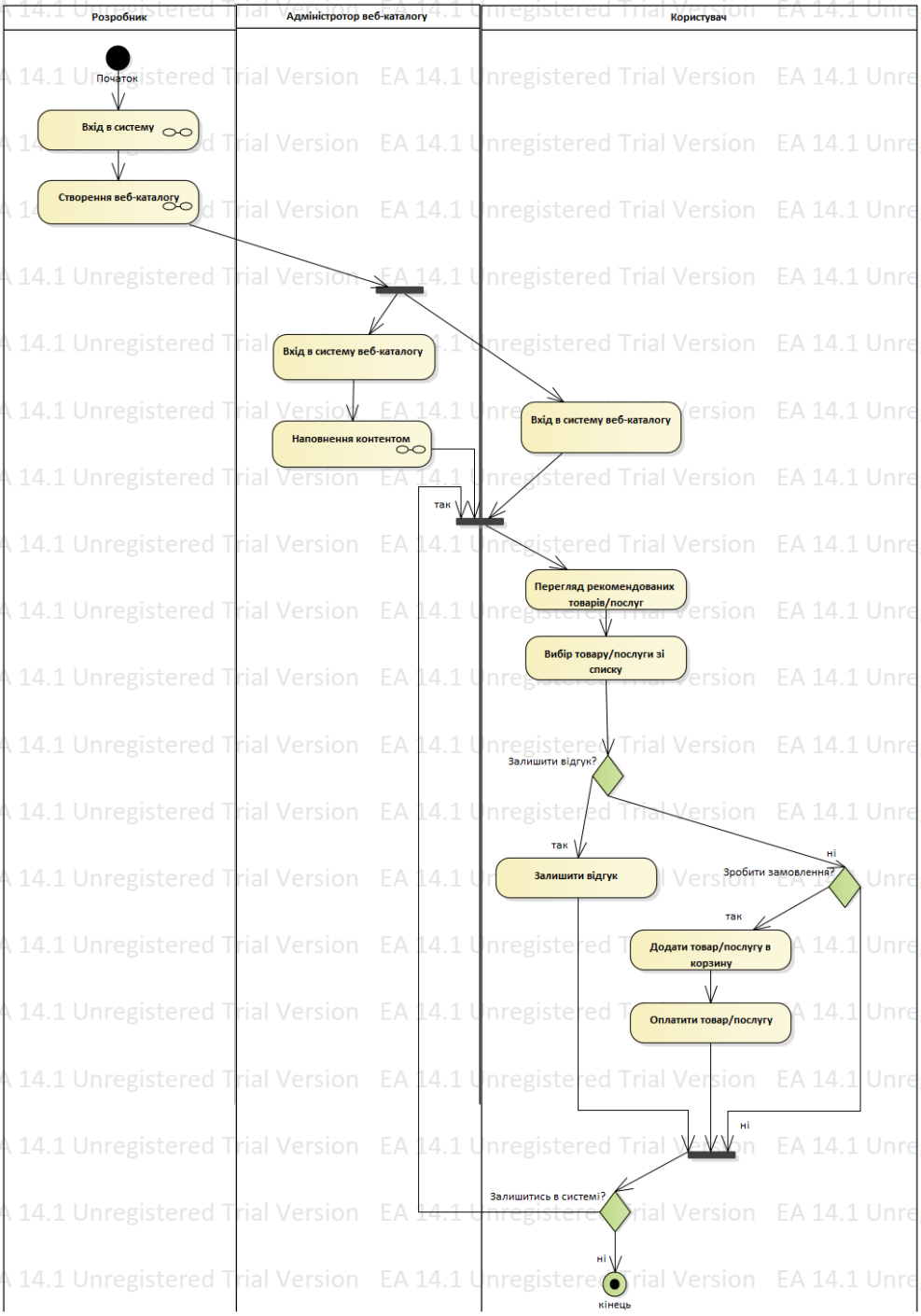


Рис. 2. Use-case діаграма

Основні дії акторів проілюстровані на діаграмі діяльності (рисунок 3).



Висновок

Для вирішення актуальних проблем, що виникають у сфері продажів було сформульовано цілі та задачі, проаналізовано існуючі рішення, запропоновано власну розробку, що відрізняється від аналогів включенням до функцій каталогів послуг та товарів рекомендаційної підсистеми.

Список літератури

1. Дино Эспозито. Разработка современных веб-приложений: анализ предметных областей и технологий. – Москва: Диалектика-Вильямс, 2017. – 464 с.

УДК 004.021

ДОРОШЕНКО А.В.,
КОЧУБЕЙ І.Ю.,
ЖУРАКОВСЬКА О.С.

АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРОЦЕСУ КЛАСТЕРИЗАЦІЇ КОРИСТУВАЧІВ І ПОЗИЦІЮВАННЯ ПОСЛУГ ТА ТОВАРІВ

В даній статті буде розглянуте актуальне питання позиціювання послуг або товарів, а також сегментації користувачів. Запропоноване алгоритмічне забезпечення побудоване шляхом поєдняння двох алгортмів: колаборативної фільтрації і кластеризації методом k-середніх.

КАТАЛОГ, КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, ПОСЛУГИ, КЛАСТЕРНИЙ АНАЛІЗ

This article will address the actual issue of positioning for different products or services and segmentation of users. This algorithm will be implemented by combining two algorithms: collaborative filtration and clustering using the k-medium method.

CATALOG, COLLABORATIVE FILTERING, GOODS, CLUSTER ANALYSIS

Вступ

В сучасному житті все більше і більше стає актуальне питання знаходження системи, яка зможе забезпечити підтримку діяльності в сфері торгівлі та послуг.

Після довгого аналізу ринку стало зрозуміло, що однією з основних проблем є створення оптимізованого каталогу продукції, що зможе значно пришвидшити доступ до каталогів та послуг. Для реалізації потрібно сегментація ринку, розробленню різних маркетингових заходів та знаходження категорії користувачі.

Для вирішення цих питань будуть розглянуті методи кластеризації та спроби поєднання з колаборативною фільтрацією.

Постановка задачі

Для того, щоб можна було створити сервіс для продажу товарів та послуг, було створено базу з користувачів, які виражені профілем поведінки у цій системі. Цей профіль можна задати множиною атрибутів, які описують користувачів та дії, які можуть виникають при роботі в системі.

Після того, як була отримана інформація від користувачів потрібно об'єднати їх в кластери за спільними характеристиками.

Після чого видати їм рекомендацію, тобто список товарів або послуг, які їх можливо зацікавлять.

Аналіз досліджень

Було розглянуто декілька олгоритмів кластеризації та класифікації.

Колаборативна фільтраця простий в реалізації метод, але він досить ефективний.

Можна побачити, що серед багатьох методів класеризації, метод k-середніх також простий, але точність даного методу велика.

Важливо замітити, що метод k-середніх має недолік, а саме необхідність вказування кількості кластерів.

Методи кореляції Пірсона та асоціативних правил не вирішує питання холодного старту хоч є прості в реалізації.

Метод TF-IDF не має такої проблеми, але він складний в реалізації.

Потрібно визначити список альтернатив а також мати експертні оцінки за множиною критеріїв.

Згідно [2] проаналізуємо, що можна включити до множини альтернатив базові алгоритми кластеризації:

- TF-IDF;
- SVD метод;
- Найближчий сусід;
- Використання кореляції Пірсона в колаборативній фільтрації;
- кластеризація k-середніх.

Оцінювання альтернатив можна здійснити за деякими критеріями:

- точність;
- ефективність;
- ціна реалізації;
- взаємозв’язок з різними алгоритмами;
- принцип роботи.

Після того, як було оцінено альтернативи за критеріями, був використаний метод ELECTRE для знаходження найкращих

алгоритмів. Метод показав, що це є колаборативна фільтрація з використання коефіцієнта Пірсона а також кластеризація методом k-середніх.

Кластеризації це процес розбиття векторів характеристик на групи. Даний процес використовує подібність між об’єктами. Метод k-середніх є дуже простий в реалізації, але потрібно задати кількість кластерів, на які будуть розбиті всі користувачі.

Перший крок алгоритму – це розбиття всіх користувачів на будь-які кластери. Вираховується центр мас для кожного кластеру. Далі потрібно обрахувати відстань від центра мас до кожного об’єкта. Після чого всі об’єкти перерозподіляються ко кластерам так, щоб між центром мас та об’єктом відстань була найменшою. Потім вираховується знову відстань від центра мас і до кожного об’єкту. Алгоритм припиняє свою роботу після того, як об’єкти не будуть змінювати кластер.

Для даного алгоритму існують метрики для того, щоб обрахувати відстані між об’єктами та центром мас:

- відстань Чебишева
- квадрат евклідової відстані;
- евклідова відстань;
- манхеттенська» відстань;
- степенева відстань.

Для реалізації алгоритму була взята евклідова відстань (1)

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2} \qquad 1)$$

Наведемо приклад кластеризації на рис.1.

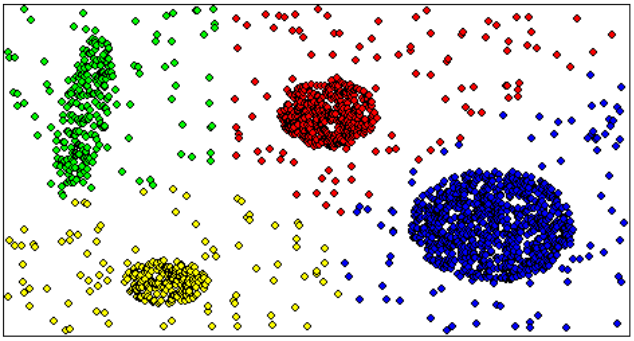


Рис. 1. Кластеризація у двовимірному просторі

На вхід алгоритму подається кількість кластерів а також множина об’єктів. Важливо помітити, що кластеризації проводиться у R^n просторі. Розмірність простору – це кількість характеристик, які є в об'єкта.

На прикладі книгарні можна визначити деяку множину характреистик:

- стать;
- вік;
- місце проживання;
- кількість переглядів книг жанру «Зарубіжна література»;
- кількість переглядів книг жанру «Психологія».

Після того, як були кластеризовані користувачі, вступає в дію алгоритм колаборативної фільтрації. На вхід подається множина користувачі з одного кластеру та користувач, для якому потрібно дати рекомендацію.

Висновок

Отже в даній статті було проаналізовану важливу проблему сегментації об’єктів, в нашому випадку користувачів системи. Використання колаборативної фільтрації разом з алгоритмом кластеризації користувачів методом k-середніх дало виграш в часі. Для вирішення задачі кластеризації користувач було розроблене алгоритмічне забезпечення. Після чого алгоритм колаборативної фільтрації давав рекомендацію користувачам в системі.

Список літератури

1. Шитиков В.К., Мастицкий С.Э. Классификация, регрессия и другие алгоритмы Data Mining с использованием R.: електронна книга,
2. Литвак Р. Б., Дорошенко А. В. Метод ELECTRE як інструмент вибору алгоритмів системи формування пропозицій користувачам книгарні. *Підсумки розвитку наукової думки: 2018*: зб. наук. праць «ΛΌГОΣ» з матеріалами міжнар. наук.-практ. конф., м. Івано-Франківськ, 5 грудня, 2018 р. Вінниця : ГО «Європейська наукова платформа». 2018. Т.4. с128. С. 88-95.

УДК 004.8

РОМАНЕНКО Л.А.

ВИКОРИСТАННЯ МАТЕМАТИЧНОГО АПАРАТУ ДИФЕРЕНЦІАЛЬНОЇ ПРИВАТНОСТІ У РОЗПОДІЛЕНОМУ МАШИННОГО НАВЧАННЯ ДЛЯ ПОРТАТИВНИХ ПРИСТРОЇВ

Коли дані користувача, такі як аудіо, фото, відеосигнали, і т.п., використовуються для навчання алгоритмів машинного навчання, забезпечення конфіденційності користувачів повинна бути розглянута в першу чергу до запуску моделі в роботу. В даній роботі описується математичне та програмне забезпечення для системи, що включає розподілене машинне навчання та зберігаючому конфіденціальність механізму збору даних з датчиків портативних пристроїв на базі математичного апарату диференціальної приватності. Система дозволяє пристроям Android спільно вивчати загальну модель класифікатора з розподілених даних з диференціальної конфіденційністю, використовуючи варіант методу стохастичного градієнтного спуску.

КЛЮЧОВІ СЛОВА: РОЗПОДІЛЕНЕ МАШИННЕ НАВЧАННЯ, ДИФЕРЕНЦІАЛЬНА ПРИВАТНІСТЬ, КРАУДСЕНСІНГ.

When user data, such as audio, photo, video, etc., is used to teach machine learning algorithms, user confidentiality should be considered first and foremost before the model is launched into operation. This paper describes the mathematical and software for the system, which includes distributed machine learning and the confidential nature of the mechanism for collecting data from sensors of portable devices based on the mathematical apparatus of differential privacy. The system allows Android devices to jointly study the overall differential classifier model with differential confidentiality using the stochastic gradient descent method option.

KEYWORDS: DISTRIBUTED MACHINE LEARNING, DIFFERENTIAL PRIVACY, CROWDSENSING

1. Вступ

Аналіз величезних даних, що генеруються портативними пристроями, без централізованого зберігання даних і порушення конфіденційності користувачів може допомогти у вирішенні багатьох проблем. Даний процес можливий через в'язку технологій таких як розподілене машинне навчання, збір даних з портативних пристроїв та механізму збереження конфіденційності – диференціальної приватності.

Система дозволяє Android-пристроям вивчати загальну модель класифікатора з диференціальною конфіденційністю шляхом вирішення задачі розподіленої оптимізації:

$$\min_{w\in W} f(w) = \min_{w\in W} \frac{1}{M}\sum_{i=1}^{M} f_i(w) \quad (1)$$

де w - загальний вектор параметрів, а$f_i(w)$ - емпіричний ризик пристрою i:

$$f_i(w) = \frac{1}{|Z^i|}\sum_{(x,y)\in Z^i} l(h(x;w),y) \qquad (2)$$

Де Z^i є н.о.р.в.в. (незалежні однаково розподілені випадкові величини) навчальний датасет пристрою i, з набором даних $Z^1, \dots, Z^M$ також від M н.о.р.в.в.-пристроїв. Незважаючи на те, що було проведено багато досліджень в області розподіленої оптимізації та навчання, в тому числі аналогічного середовища навчання для інтелектуальних пристроїв [4], дана робота унікальна тим, що являє собою розподілене машинне навчання з акцентом на конфіденційність.

Ітеративна оптимізація таких цілей, розподілених по декількох агентам, вивчалася багатьма дослідниками (наприклад [1, 2, 3, 4].) Дана стаття фокусується на напів-синхронній моделі, яка складається з декількох пристроїв і центрального сервера-координатора з обмеженими затримками зв'язку (див. Рис. 1.).

Представляється варіант оптимізації стохастичного градієнтного спуску (СГС), який використовує міні-пакети на стороні пристрою і на стороні сервера, а також дозволяє локальне оновлення ваг на пристроях. Кожен пристрій обчислює і передає вектор вагових коефіцієнтів w після ряду локальних оновлень, потім сервер зберігає середнє значення всіх w від декількох пристроїв. Простота оптимізації на основі градієнта дозволяє виконувати таку процедуру на інтелектуальних пристроях з обмеженими ресурсами.

Останнім часом багато дослідників включили механізми конфіденційності в стохастичну оптимізацію на основі градієнтного спуску [5, 6] або в рішення задач розподіленого навчання [7, 8, 9, 10],

використовуючи диференціальну конфіденційність як вимірюваної міри конфіденційності [11, 12 13]. Відповідний вибір механізму залежить від декількох цільових функцій (наприклад, ERM зі строго опуклою, диференційованою і гладкою функцією втрат [14]), гранулярної конфіденційності (наприклад вибірка, людина або група) і припущення проти).

У цій роботі використовується адитивний шум для «дезінфекції зв'язку» між пристроєм та сервером.

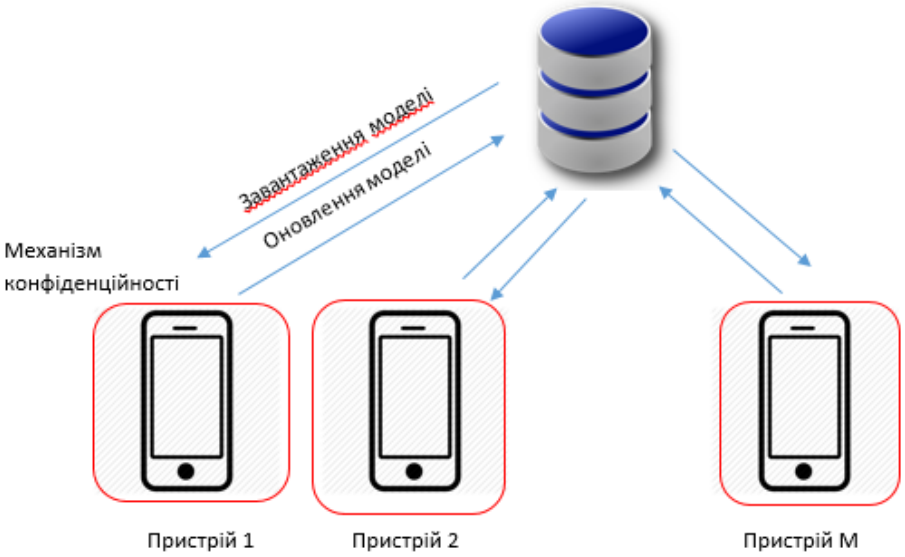


Рис. 1 Схема розподіленого навчання

2. Спотворення даних адитивним шумом

Адитивний шум дискретизується незалежно і додається до параметрів, що передаються з пристрою на сервер. Два часто використовуваних розподіли – це гауса і лапласа (6) дистрибутиви, які реалізовані в роботі. З огляду затрат ϵ на конфіденційність і максимальну кількість передач даних T_d, простота механізму полягає в використанні н.о.р.в.в. шуму для кожної передачі даних. Однак шум може бути обраний для геометричного зменшення [8], зменшення пропорційно швидкості навчання або вибірки з випадкового процесу, в якому послідовність шуму не є незалежною [17]. Вибір оптимальної послідовності шуму з найменшими втратами корисності залишається відкритою проблемою.

3. Розподілене машинне навчання

В даній роботі використовується варіант стандартного стохастичного градієнтного спуску, щоб вирішити (1).

$$w(t+1) = w(t) - \eta(t)\nabla f_i(w(t)) \quad (3)$$

По-перше, використовуються міні-пакети як на стороні пристрою, так і на стороні сервера. Міні-пакет на стороні пристрою грає важливу роль в скороченні кількості раундів зв'язку. У середовищі, де зв'язок дорогий, збільшення розміру міні пакета на стороні пристрою B_d дозволяє виконувати більше роботи на кожному пристрої, знижуючи частоту зв'язку. Міні пакет на стороні сервера грає іншу роль. Міні-пакет на стороні сервера знижує дисперсію шумних ваг від кількох пристроїв, а також пом'якшує негативні наслідки затримки зв'язку [3]. По-друге, кожен пристрій виконує самооновлення (3) і відправляє поточні параметри w на сервер, замість того, щоб відправляти градієнт і покладатися на сервер для виконання оновлення (3). Аналогічно міні-пакету пристроя, локальні поновлення можуть знизити частоту зв'язку з сервером і тримати пристрій, в процесі покращення своєї моделі, коли знаходиться в автономному режимі.

4. Математичний апарат диференціальної приватності

Рандомізований алгоритм, який приймає дані D в якості вхідних даних і виводить функцію f, називається диференціально закритим, якщо

$$P(f(D) \in S) \leq e^{\epsilon} P(f(D`) \in S) \quad (4)$$

для всіх вимірних S ⊂ T вихідного діапазону і для всіх наборів даних D і D`, що відрізняються одним елементом, позначених D~D`. Тобто, навіть якщо зловмисник знає весь набір даних D, за винятком одного елемента, він не може дізнатися більше про невідому частину з результату (виводу) f алгоритму. Коли алгоритм виводить дійсний вектор $f \in R^D$, його глобальна чутливість [12] може бути визначена як

$$S(f) = \max_{D \sim D`} \|f(D) - f(D`)\| \quad (5)$$

де $\|\cdot\|$ є нормою. Важливим результатом з [12] є те, що вектор-вихідних даних f з чутливістю S (f) можна зробити - диференційно-приватним шляхом спотворення f з додаванням η вектора адитивного шуму, щільність якого дорівнює

$$P(\eta) \propto e^{-\frac{\epsilon}{S(f)}\|\eta\|} \quad (6)$$

5. Використання диференційної приватності

Обчислюється чутливість кожного раунду комунікацій. Без обмеження спільності припускається, що D і D' різні тільки для пристрою 1 і однакові для інших пристроїв. Особисті дані пристрою 1 можуть довільно відрізнятися в D і D'. Припустимо, що пристрій 1 запускається з вагами $w(t, 0)$, та виконує локальні поновлення для S_d раз

$$w(t, s+1) = w(t, s) - \eta(t, s) \nabla f_i\big(w(t, s)\big), \quad s = 0, \dots, S_d - 1 \quad (7)$$

де $w(t, 1), \dots, w(t, S_d)$ - послідовність проміжних вагових значень. Після локальних оновлень S_d отримуємо

$$w(t, S_d) = w(t, 0) - \sum_{s=0}^{S_d - 1} \eta(t, s) \nabla f_i(w(t, s)) \quad (8)$$

і, отже, чутливість ваг відправляється на сервер $S(w(t, S_d))$ є

$$\max_{D \sim D`} \|w_D(t, S_d) - w_{D`}(t, S_d)\| = \max_{D \sim D`} \Big\| \sum_{s=0}^{S_d - 1} \eta(t, s) \big[\nabla f_i\big(w_D(t, s)\big) - \nabla f_i`\big(w_{D`}(t, s)\big)\big] \Big\| \quad (9)$$

Для неадаптивної і спадної швидкості (t, s) маємо

$$S\big(w(t, S_d)\big) \leq S_d \eta(t, 0) S(\nabla f_i), \quad \text{де}$$

$$S(\nabla f_i) = \max_{D \sim D`} \big\| \nabla f_i\big(w_D(t, s)\big) - \nabla f_i`\big(w_{D`}(t, s)\big) \big\| \quad (10)$$

є чутливістю градієнта.

Слід зазначити, що якщо сервера є довіреними, то чутливість може бути зменшена, щонайменше, в 1 раз = Bs, оскільки противник (тобто інший пристрій) бачить тільки усереднений параметр (вектор) пристроїв Bs. Крім того, якщо гранулярність конфіденційності, яка розглядається, є ще однією вибіркою замість всіх N вибірок пристроїв, у нас є додатковий 1 = N фактор зниження чутливості. Таке зниження чутливості може зробити навчання, що зберігає конфіденційність – практичним.

Висновки

У цій статті було описано загальну схему побудови машинного навчання зі збереженням конфіденційності для пристроїв Android, яка мінімізує емпіричні втрати різних типів, використовуючи варіант стохастичного градієнтного спуску, який використовує міні-пакети на стороні пристрою і на стороні сервера і дозволяє локальні поновлення ваг. Зв'язок між пристроєм і сервером спотворюється адитивним шумом, щоб відповідати різним критеріям конфіденційності.

Список літератури

1. Ж.Н. Цицікліс Д.П. Берцекас, М. Атанс, “Розподілені асинхронні детерміновані та стохастичні алгоритми оптимізації градієнтів” Конференція - в американському контролі, 1984, с. 484–489.
2. О. Декель, Р. Гілад-Бахрах, О. Шамір і Л. Сяо, “Оптимальне розподілене онлайн-прогнозування”, у Proc. Міжнародна конференція з машинного навчання, 2011.
3. Г.Б. МакМахан, Е. Мур Д. Рамаж, та ін., "Федеративне навчання глибоких мереж, використовуючи моделювання усереднення", arXiv препринт arXiv: 1602.05629, 2016.

4. С. Сонг, К. Чаудхурі і А.Д. Сарват, “Стохастик градієнтний спуск з диференційно-приватними оновленнями» з глобальної конференції з обробки сигналів та інформації, 2013 IEEE. IEEE, 2013, с. 245–248.
5. Р. Бассіль, А. Сміт і А. Тхакурта, “Приватна емпірична мінімізація ризику: ефективні алгоритми і тісні межі похибки”, в фундаментах комп'ютерних науках, IEEE 55-й щорічний симпозіум. IEEE, 2014, с. 464–473.
6. А. Райкумар і С. Агарвал, “Алгоритм диференційно приватного стохастичного градієнтного спуску для багатопартійності класифікації”, міжнародна конференція з штучного інтелекту та статистики, 2012, с. 933–941.
7. З. Хуан, С. Мітра та Н. Вайдя, “Диференціально приватна розподілена оптимізація”, у працях 2015 Міжнародна конференція з розподілених обчислень і мереж. ACM, 2015, с. 4.
8. Дж. Хамм, А. Чемпіон, Г. Чен, М. Бєлкін, Д. Сюань: “Crowd-ML: фреймворк для конфіденційного навчання групи пристроїв», у працях 35-ї міжнародної конференції IEEE з розподілених обчислювальних систем (ICDCS). IEEE, 2015.
9. Дж. Хамм, П. Цао і М. Бєлкін, «Конфіденційне навчання для розподілених даних», у працях 33-ї міжнародної конференції з машинного навчання, 2016, с. 555–563.
10. Ч. Дворк и К. Нісим, "Захист конфіденційності даних на вертикально розділених базах даних", досягненнях в криптології. Springer, 2004, с. 528–544.
11. Ч. Дворк, Ф. МакШеррі, К. Ніссім і А. Сміт, "Калібрування шуму до чутливості при приватному аналізі даних", Теорія криптографії, с. 265–284. Springer, 2006.
12. Ч. Дворко, “Диференціальна конфіденційність”, в автоматах, мовах та програмуванні, с. 1–12. Springer, 2006.
13. К. Чаудхурі, К. Монтелеоне, и А. Д. Сорват, "диференційно приватна емпірична мінімізація ризику", Журнал дослідження машинного навчання, вип. 12, с. 1069–1109,2011.
14. Дж. Дучі, Е. Хазан, Я. Сінгер,“Адаптивні субградієнтні методи для онлайн-навчання та стохастичної оптимізації”, Journal of Machine Learning Research, vol. 12, ні. Jul, pp. 2121–2159, 2011.
15. Т. Тіельман і Г. Хінтон, “Coursera: нейронні мережі для машинного навчання, лекція 6.5” 2012.
16. Р. Холл, А. Рінальдо і Л. Вассерман, “Диференціальна конфіденційність для функцій і функціональних даних” Журнал дослідження машинного навчання, вип. 14, №. Лют., С. 703–727, 2013.
17. Р. Шокрі і В. Шматиков, «Глибоке збереження конфіденційності навчання» у працях 22-го ACM SIGSAC Конференція з безпеки комп'ютерів та комунікацій. ACM, 2015, с. 1310–1321.
18. М. Абаді, А. Чу, І. Гудфеллоу, Г.Б. McMahan, І. Міронов, К. Талвар, Л. Чжан, “Глибоке навчання з диференційованою конфіденційністю», у працях 2016 року ACM SIGSAC Конференція з компютерної безпеки та комунікацій. ACM, 2016, с. 308–318.
19. С. Дворк і А. Рот, “Алгоритмічні основи диференціальної конфіденційності” Основи та тенденції в теоретичній інформатиці, том. 9, №. 3 4, с. 211–407, 2014.

УДК 519.876.5

АНИЩЕНКО К.М.
ЖДАНОВА О.Г.

ДОСЛІДЖЕННЯ СТРАТЕГІЙ РОЗПОДІЛУ НАГОРОД В ДЕЦЕНТРАЛІЗОВАНИХ МЕРЕЖАХ

У статті розглядається властивості розподілу нагород у децентралізованих мережах. Розглядається така стратегія розподілу, яка б забезпечила стимулювання користувачів у відповідності до необхідних властивостей мережі. За допомогою моделювання на різних типах мереж, близьких за характеристиками до реальних, проводиться аналіз життєздатності такої стратегії.
Ключові слова: децентралізовані мережі, розподіл нагород, стратегія, моделювання.

The article deals with the distribution of awards in decentralized networks. A distribution strategy is considered that would provide incentives for users in accordance with the required network properties. By simulating on different types of networks, close to the characteristics of the real, an analysis of the viability of such a strategy.
Key words: decentralized networks, reward distribution, strategy, modeling.

1. Вступ

В сучасному світі ми стикаємося з великою кількістю мереж різного типу: соціальні мережі, мережі елекротропередач, мережі постачання, та ін. Значна їх кількість має один або декілька центрів, з яких ця мережа керується чи регулюється. Але з розвитком технічного прогресу з'являється можливість уникнути участі третьої сторони у взаємодії між учасниками мережі – так з'являються децентралізовані мережі, де усі правила роботи мережі закладені в її структурі та протоколах.

Важливо розуміти, які види децентралізованих мереж існують у сучасному світі, а також що робить їх і децентралізованими, і ненадійними (trustless) одночасно. Базовою технологією, на яку спираються цифрові валюти, є блокчейн. Простіше кажучи це просто послідовно оновлюваний запис обміну інформацією у мережі персональних комп'ютерів. Цією мережею ніхто не володіє, тому вона є децентралізованою і її складно зламати. Такі мережі можуть також бути використані поза сферою фінансів (наприклад, вибори). Ненадійною її робить те, що будь-хто може додавати в блокчейн інформацією, тому існують певні протоколи згоди (consensus protocols), які дозволяють підтвердити коректність операцій: proof-of-work, proof-of-stake, byzantine fault tolerance, delegated proof-of-stake. Це дозволяє обійтися без будь-якої централізованої форми прийняття рішень [1].

Але відсутність у таких мережах регулюючої сторони, яка має контроль над усією мережею, несе за собою інший клас ризиків. Коли діяльність користувачів ніхто не контролює, їх хаотична поведінка може дестабілізувати усю мережу. Інтереси користувачів можуть не співпадати або навіть протирічити інтересам самої мережі, що робить таку мережу нежиттєздатною, і в перспективі це призведе до того, що більшість користувачів просто перестануть нею користуватися.

Тому постає проблема у стимулюванні користувачів таким чином, щоб корисна для них поведінка ставала запорукою існування мережі в цілому. Необхідно якомога більше зблизити інтереси користувачів та мережі, а також зменшити вплив деструктивної поведінки учасників мережі.

2. Модель мережі

Будемо розглядати модель системи, яка базується на використанні понять теорії графів. В цій моделі вершини графу відповідають користувачам мережі, дуги відображають взаємовідношення між її користувачами, а нагорода, яка надходить до мережі, поширюється дугами графу.

Користувачі мережі розділяються на 2 типи: *автори* та *підписники*. Автори є

учасниками мережі, які виробляють певний контент, що може бути цікавий підписникам. Підписники можуть формувати зв’язки у мережі, підписуючись на авторів, які продукують цікавий їм матеріал. Також деякі підписники можуть бути джерелами винагороди, тобто через них у мережу буде потрапляти нагорода, яку необхідно розподілити відповідно до існуючих зв’язків.

Стратегія розподілу нагород у мережі має бути такою, що буде призводити до отримання учасниками мережі, що поводять себе корисним для мережі чином, більшої частки винагороди у порівнянні з іншими.

3. Гіпотези щодо мережі

Потрібно спочатку визначити критерії, за якими ми зможемо зробити висновок про те, чи відповідає певна розглядувана стратегія нашим вимогам. Для цього були сформульовані гіпотези, виконання яких було б бажано отримати у результаті моделювання. Розділимо гіпотези за типом вузлів, до яких вони відносяться.

Гіпотези щодо авторів

Логічним є припущення, що чим більше у автора є підписників, тим більше нагороди він повинен отримувати (велика кількість зв'язків говорить про популярність та більший інтерес з боку аудиторії). Залежність повинна бути лінійною або логарифмічною, щоб на початкових етапах невеликі прирости користувачів давали більший приріст нагороди. А чим більш популярним стає автор, тим простіше йому наростити нову аудиторію, і відносно невеликі прирости вже не повинні давати такий значний результат, як раніше.

Інше припущення базується на тому, що ранні за часом («старі») зв'язки повинні мати більше значення. Це означає, що автор продовжує залишатися цікавим протягом довгого часу. Це буде стимулювати авторів зберігати підписників, утримувати їх підтримкою якості свого контенту на належному рівні.

Гіпотези щодо підписників

Підписників потрібно стимулювати підписуватися на більш цікавих і перспективних авторів. Якщо вимірювати цей показник тільки кількістю зв'язків, то буде виділена якась множина авторів, на яку буде вигідно підписуватися всім, і з часом такий перекіс буде ставати більш явним, і уся мережа зосередиться навколо них. Цьому можна завадити, враховуючі час підписок таким чином, щоб ранні підписки давали більше нагороди. Це спонукатиме підписників підтримувати тих авторів, які можуть стати популярні в майбутньому. І навпаки, треба деяким чином давати менше нагороди новим підписантам найбільш популярних артистів. Бо їх привабливість є очевидною, і нові його підписники не дають багато нової інформації про цікавих авторів. Це буде більше стимулювати підписників підтримувати перспективних, але поки непопулярних авторів. Тобто давати більшу нагороду першим підписникам, які підтримували конкретного успішного автора до того, як він став загальновідомим (малочисельні фани нових авторів в подальшому повинні мати переваги при отриманні винагороди – відносна її кількість повинна збільшуватись із зростанням популярності автора).

Чим більше зв'язків формує підписник, тим менш значущими вони повинні бути. Потрібно стимулювати підписників фокусуватися на найбільш цікавих авторах, а не просто підписуватися на якомога більшу їх кількість. Така поведінка має призводити до зменшення отримуваної нагороди. Для додаткового гарантування відсутності підписників з надвеликою кількістю зв’язків можна просто обмежити кількість можливих зв'язків, доступних підписнику. Також можна збільшувати цей ліміт з часом, щоб список цікавих авторів міг бути розширений тільки в майбутньому, а не одразу після приєднання до мережі.

Наявність єдиного зв'язку для підписника також не є дуже корисною, тому що вона не створює нових шляхів між іншими вузлами, тобто не дає нову інформацію про близькість вузлів. Тому потрібно накладати також і обмеження на мінімальну кількість зв'язків.

4. Види мереж, які використані для моделювання стратегій

Випадкові мережі

Найбільш простим та зрозумілим варіантом побудови мережі є випадковий вибір зв’язків. Створюється необхідна кількість вузлів, після чого з певною ймовірністю обираються зв’язки між ними. Ця ймовірність буде характеризуватися

щільністю графу – відношенням кількості існуючих зв'язків у мережі до загальної кількості усіх можливих зв'язків. Така модель має назву модель Ердоса-Рені [2]. Далі такі графи називатимемо RNG.

Хоча таку мережу зручно використовувати для перевірки загальних закономірностей, вона не зовсім підходить для перевірки роботи реальних мереж. Тому що в дійсності люди не формують рівномірні мережі. В такій мережі коефіцієнт кластеризації прямує до нуля з ростом мережі.

Реальні мережі мають деякі спільні характеристики. Декілька статей, наприклад [3, 4], дають огляд цих властивостей:

– деревоподібні: кількість ребер росте лінійно від кількості вузлів;

– домінуюча компонента: мережа може бути розділена на декілька компонент, але одна з них представляє більшу частину мережі.

– властивість «тісного світу»: діаметр мережі оцінюється як $O(\log N)$, тобто пара вузлів однієї компоненти з'єднані коротким шляхом.

– безмасштабні (scale-free): розподіл степенів вершин відповідає ступеневому закону.

– коефіцієнт кластеризації великий: в реальних мережах він більший ніж у випадкових, демонструючи їхню тенденцію до групування.

Безмасштабні мережі

Безмасштабні мережі (scale-free network) – такий тип мереж, у яких розподіл степенів вершин асимптотично підкоряється ступеневому закону [5]. Тобто кількість вершин $P(k)$ зі степенем k визначається так:

$$P(k) \sim k^{-\gamma}$$

де γ – параметр, який зазвичай знаходиться у межах від 2 до 3.

Багато реальних мереж добре моделюються за допомогою безмасштабних мереж: соціальні мережі, графи цитування, сторінки в Інтернеті, комунікаційні, біологічні та багато інших.

Найбільш помітною характеристикою безмасштабних мереж є наявність вершин з дуже великою кількістю зв'язків. Такі вершини називають концентраторами (hubs).

Іншою важливою характеристикою є розподіл коефіцієнту кластеризації, що зменшується зі збільшенням степені вершин та також підкоряється ступеневому закону. Тобто вершини з невеликою кількістю зв'язків формують щільні підграфи, що поєднуються між собою за допомогою концентраторів. Це відповідає реальній поведінці соціальних мереж, коли люди формують невеликі спільноти, у яких кожен знає майже всіх, а також є деякі люди, які належать до великої кількості різних спільнот (відомі актори, політики і т.п.)

Мережа «тісний світ»

Простий випадковий граф характеризується короткими відстанями між вузлами, але має малий коефіцієнт кластеризації. Аналіз реальних мереж показує, що вони мають значно більший коефіцієнт кластеризації та менші відстані між вузлами.

Для створення мереж з такими властивостями існує модель Воттса-Строгаца [6]. Такі мережі мають назву «тісний світ», бо у них відстані між вузлами є невеликими у порівнянні із розміром мережі. Далі будемо називати такі мережі SWG (Small world graph).

Спочатку із заданого набору вузлів будується кільце, де кожен вузол з'єднується з певною заданою кількістю сусідніх. Потім для кожного вузла розглядаються ребра, які йдуть від нього за годинниковою стрілкою, та з певною заданою ймовірністю p кінець цього ребра може бути змінений на рівноймовірно обраний вузол серед усіх.

Основним недоліком такої мережі є нереальні степені вершин. В реальності мережі дуже неоднорідні у степенях, і для створення мереж з цією властивістю існують окремі моделі, які будуть описані далі.

Оскільки в нашому випадку граф є дводольним і в ньому не може бути об'єднана будь-яка пара вершин, то для того, щоб застосувати цей вид мережі, в модель мережі необхідно внести певні корективи. Будемо діяти наступним чином. Якщо кількість авторів та підписників співпадає, то ми можемо спочатку розмістити їх через одного та отримати бажане коло. В реальних мережах кількість авторів буде меншою за кількість підписників, і ми замість одного вузла для підписника у кільці будемо розміщувати декілька підписників шарами, рівномірно розподіляючи їх по колу.

Далі все виконується аналогічно діям, описаним для загальної моделі.

Мережі переважного приєднання

Багато мереж реального світу характеризуються тим, що є невелика кількість вузлів, які мають надвелику кількість зв'язків порівняно з іншими. Тобто існують певні хаби (центри груп), які концентрують навколо себе багато інших учасників. Для відтворення такої властивості існує модель Барабаши-Альберт [7].

Принцип переважного приєднання полягає у тому, що вузли з більшою кількістю зв'язків мають більшу ймовірність на формування нових зв'язків. У реальних соціальних мережах можна провести аналогію з відомими людьми, які вже мають багато зв'язків. Коли новий учасник приєднується до мережі, він більш ймовірно приєднається до більш відомих учасників, ніж до якихось невідомих. Іноді це також називають ефектом Матфея або «багаті стають багатшими». Далі такі мережі будемо називати PAG (Preferential attachment graph).

Для побудови мережі спочатку виділяється невелика множина хоча б з двох вузлів, у кожного з яких є хоча б по одному зв'язку. Далі по черзі додаються нові вузли, при цьому ймовірність приєднання пропорційна до кількості зв'язків:

$$p_i = \frac{k_i}{\sum_j k_j}$$

де k_i – степінь i-го вузла, що вже є у мережі.

В нашому випадку ми будемо починати будувати модель з двох поєднаних вузлів: автора та підписника. Після чого у випадковій послідовності будемо додавати ті вузли, що залишилися, формуючи зв'язки відповідно до описаного вище алгоритму. В результаті буде невелика кількість авторів, що мають дуже багато прихильників, та багато маловідомих авторів, що в багато чому відповідає дійсності.

Побудований таким чином граф буде мати необхідний нам розподіл степенів вершин. Для логарифмічних осей ступеневий закон має вигляд прямої, що ми і бачимо на рисунку 1.

5. Розробка стратегії розподілу

В якості базової стратегії була взята та, що розглядається у нашій попередній роботі [8]. Тобто нагорода розподіляється хвилями. Коли нагорода потрапляє у певну вершину, частина її надається цій вершині, а залишок рівномірно розподіляється між її сусідами за виключенням тих, що вже приймали участь у розповсюдженні цієї частки на попередніх хвилях, і процедура повторюється далі.

Але такий розподіл не надає нам можливість стимулювати підписників формувати прийнятну кількість зв'язків. Також вона ніяким чином не враховує час. Тому необхідно розширити стратегію для виконання розглянутих вище гіпотез.

Почнемо з того, що нагорода тепер не повинна розподілятися рівномірно між сусідами. Замість цього введемо вагові коефіцієнти, пропорційно до яких буде розподілятися залишок нагороди, що виходить із вершини. Ці вагові коефіцієнти і мають залежати від характеристик вершин і для кожного з типів вершини процедура розрахунку коефіцієнтів повинна проводитись окремо.

Для врахування часу пропонується включити у вагові коефіцієнти для обох типів множник, що залежить від *давності* створення зв'язку, по якому нагорода потрапляє у наступну вершину (кількість часу між поточним моментом та моментом утворення зв'язку). Також необхідно додати множник, що залежить від кількості зв'язків. Загальний вигляд коефіцієнту вершини при розподілі винагород:

$$r(k,t) = r_1(k) \cdot r_2(t)$$

де k – кількість зв'язків у вершині, t – давність зв'язку, по якому нагорода потрапляє у вершину.

Для часу зв'язків пропонується множник у вигляді логарифму давності, щоб не створювати занадто великий дисбаланс у сторону старих зв'язків.

$$r_2(t) = \ln(t)$$

Для авторів кількість зв'язків не потрібно якось обмежувати, бо чим більше зв'язків, тим більшою є популярність, і тим більше нагороди було б логічно віддавати саме таким авторам. Тому у коефіцієнт для авторів необхідно включити множник, пропорційний до кількості зв'язків.

$$r_1(k) = Bk$$

де B – деякий дійсний коефіцієнт пропорційності.

Для підписників нам потрібно ввести коефіцієнт, який би надавав перевагу

помірному значенню кількості зв'язків та штрафував занадто малу чи велику кількість зв'язків. Таким вимогам відповідає така формула обрахунку коефіцієнтів підписників:

$$r_1(k) = \left(\frac{k}{C}\right)^D e^{-\left(\frac{k}{C}\right)}$$

де C, D – деякі дійсні коефіцієнти.

При додатних аргументах ця функція має такий вигляд.

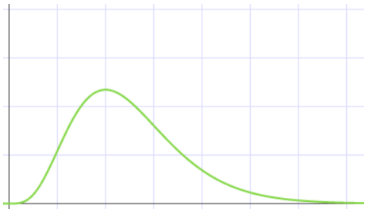


Рис. 1 – Складова коефіцієнту підписників, що залежить від кількості зв'язків

Тобто є деяка кількість зв'язків, за якої досягається максимум. Чим менше або більше є значення кількості зв'язків, тим більше такі вершини штрафуються (отримують менше винагороди). Змінюючи коефіцієнти C і D можна регулювати оптимум цієї функції. Максимум буде досягатися при кількості зв'язків, що дорівнює добутку C та D.

6. Планування експериментів

Спираючись на перелік гіпотез було запропоновано наступний план експериментів, результати яких здатні надати підстави для підтвердження чи спростування цих гіпотез. В описі експериментів будуть виділятись окремі множини вершин T, на які будуть накладатися окремі умови. Вони можуть бути як виділені серед існуючої множини вузлів, так і додані окремо.

Експерименти для перевірки гіпотез авторів

1. Виконати моделювання для випадкового графу та визначити залежність кількості отриманої винагороди від кількості зв'язків авторів. Відповідна функція повинна бути зростаючою та бути близької до лінійної або логарифмічної.

2. Виконати моделювання для випадкового графу з урахуванням часу. Множина T буде складатися з авторів, для яких час утворення усіх зв'язків у конкретного автора є однаковим, і кількість зв'язків у всіх цих авторів теж однакова, але час для авторів різний. Визначити для T залежність кількості винагороди від давності існування зв'язків. Відповідна функція повинна бути зростаючою.

Експерименти для перевірки гіпотез підписників

1. Виконати моделювання для випадкового графу та визначити залежність кількості отриманої винагороди від кількості зв'язків підписників. Для дуже малої кількості зв'язків нагорода повинна бути малою, далі вона повинна зростати, а після деякого оптимуму спадати.

2. Виконати моделювання для випадкового графу з урахуванням часу. Множина T буде складатися з підписників, для яких час утворення усіх зв'язків у конкретного підписника є однаковим, і кількість зв'язків у всіх цих підписників теж однакова, але час різний серед підписників. . Визначити для T залежність кількості винагороди від давності існування зв'язків. Відповідна функція повинна зростати лінійно чи експоненційно (на більших термінах різниця буде мати більше значення). Виконати аналіз для різної кількості зв'язків.

Кожна з вершин підписників може бути генератором винагороди. Для моделювання ми припустимо, що ймовірності того, що підписник є генератором винагороди, є однаковою для всіх підписників. Тому ми проведемо введення винагороди в усі вершини підписників по черзі. Це дасть нам змогу проаналізувати поведінку розподілу, якщо б час моделювання був нескінченним і кожен з вузлів був би генератором однакову кількість разів.

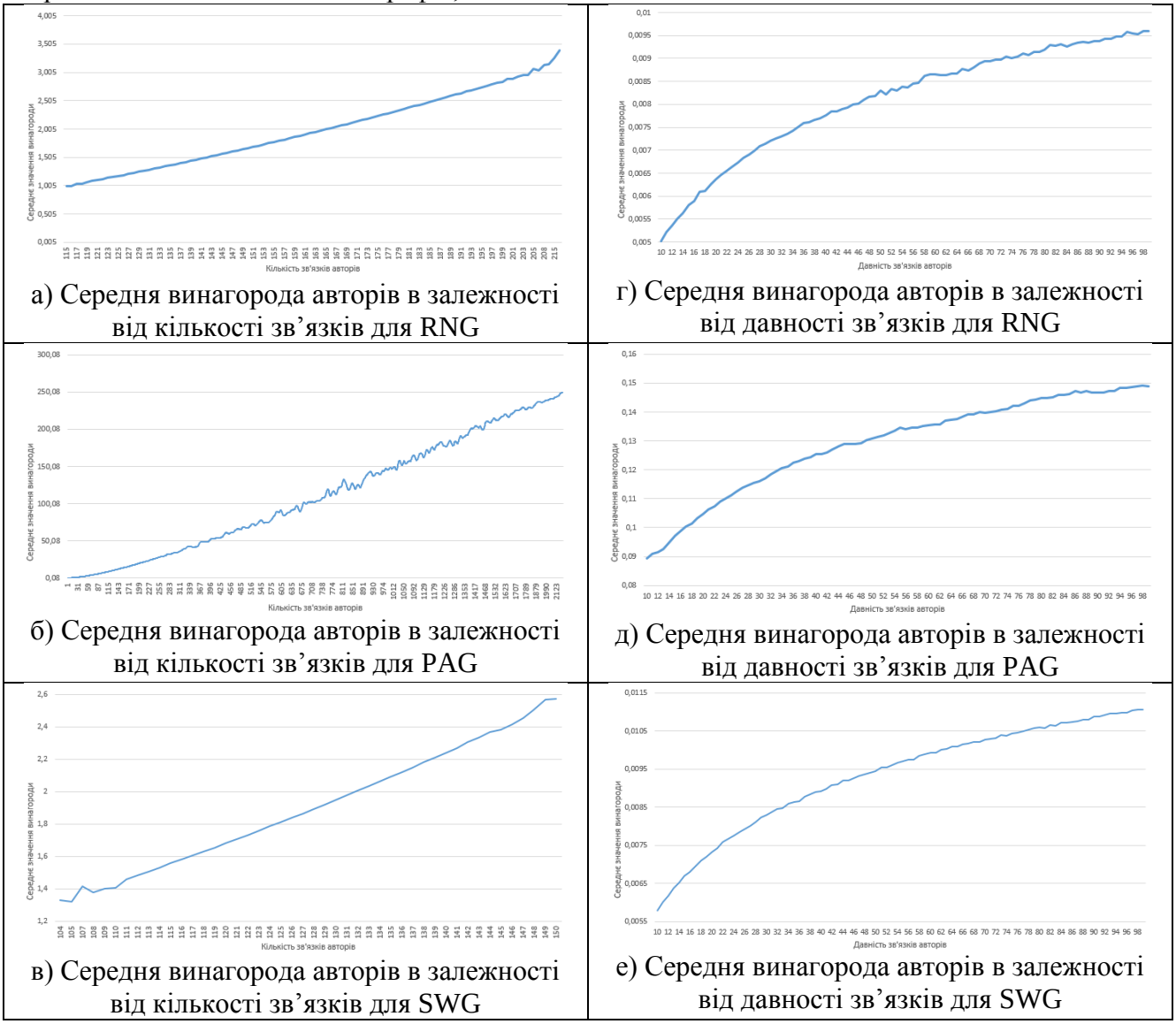
Для авторів результати можна бачити одразу. Для підписників треба виключити випадок, коли певний підписник є генератором винагороди. Для цього потрібно відняти значення винагороди, яка залишається у вершині після введення винагороди у нього. Таким чином ми отримаємо результати розподілу для кожного з підписників, які виключають його участь в якості генератора. Тобто буде враховуватись лише структура зв'язків, які сформовані у мережі.

7. Результати експериментів

Експерименти проводились для графів з наступними характеристиками: кількість вершин: 1000; кількість авторів: 200;

щільність RNG: 0.2; ймовірність p для SWG: 0.2; частка нагороди, що залишається у вершині: 0.1 для підписників та 0.2 для авторів; $B=1, C=8, D=4$; кількість прогонів: 1000.

Як видно з графіків винагороди для авторів, залежність величини нагороди від кількості зв'язків є близькою до лінійної. Це справедливо для всіх типів графів, для яких проводилось моделювання. Залежність винагороди від давності зв'язків має логарифмічний характер для всіх типів графів. Тобто властивості, закладені нами у стратегії розподілу, були збережені під час розподілу та підтверджуються результатами експериментів. Графіки отриманих залежностей для підписників представлені на рисунку 2.



а) Середня винагорода авторів в залежності від кількості зв'язків для RNG

г) Середня винагорода авторів в залежності від давності зв'язків для RNG

б) Середня винагорода авторів в залежності від кількості зв'язків для PAG

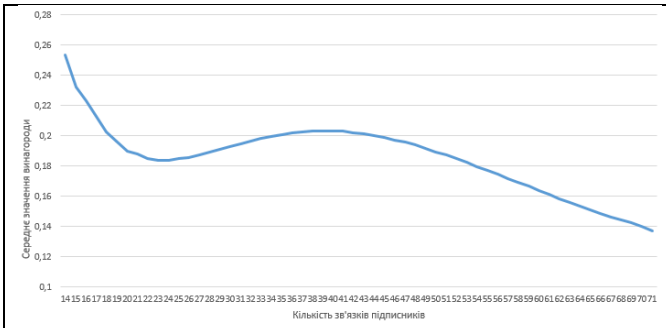
д) Середня винагорода авторів в залежності від давності зв'язків для PAG

в) Середня винагорода авторів в залежності від кількості зв'язків для SWG

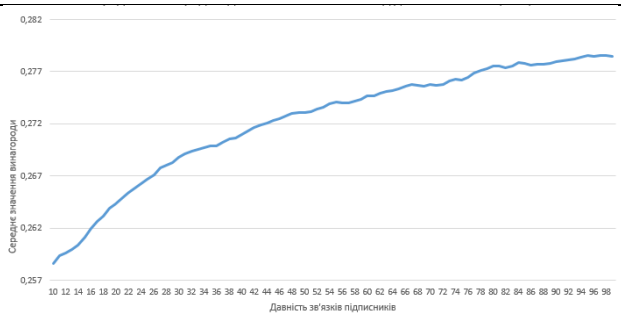
е) Середня винагорода авторів в залежності від давності зв'язків для SWG

Рис. 2 – Результати експериментів для авторів

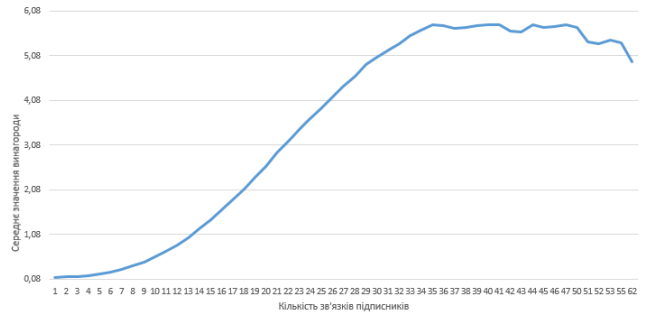
Для підписників графіки залежності від часу також підтверджують нашу вимогу на логарифмічну залежність винагороди. Але для графіків нагороди від кількості зв'язків ми бачимо деякі відхилення. Для PAG графік здебільшого повторює запитувану нами залежність, але на великих значеннях ми маємо досить значні шуми. Це спричинено тим, що навіть для великої кількості прогонів випадки, коли підписники мають певну велику кількість зв'язків у мережі досить нечасті. Для RNG та SWG таких шумів не спостерігається, але є стрибок у значеннях винагороди для найменших значень кількості зв'язків. Також для RNG є деякий зсув оптимальної кількості зв'язків вправо (за розрахунками оптимум відповідає 32 зв'язкам). Графіки отриманих залежностей для підписників представлені на рисунку 3.



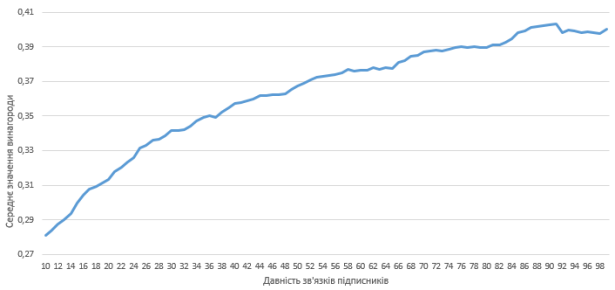
а) Середня винагорода підписників в залежності від кількості зв’язків для RNG



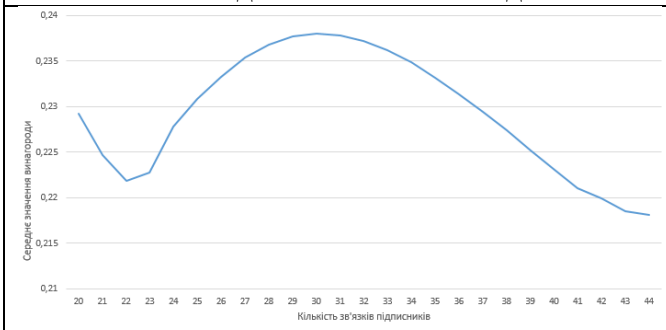
г) Середня винагорода підписників в залежності від давності зв’язків для RNG



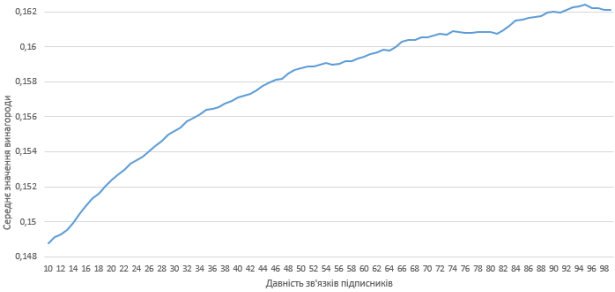
б) Середня винагорода підписників в залежності від кількості зв’язків для PAG



д) Середня винагорода підписників в залежності від давності зв’язків для PAG



в) Середня винагорода підписників в залежності від кількості зв’язків для SWG



е) Середня винагорода підписників в залежності від давності зв’язків для SWG

Рис. 3 – Результати експериментів для підписників

Висновки

В даній роботі було запропоновано стратегію розподілу нагород для децентралізованих мереж та проаналізовано результати моделювання з її використанням. Для авторів отримано бажані залежності, а для підписників спостерігаються неочікувані ефекти збільшення винагороди для малої кількості зв’язків. У подальшому планується дослідити причини таких ефектів.

Перелік посилань

1. Blockchain: Decentralized & Trustless Networks. [Електронний ресурс] // Режим доступу: https://medium.com/coinbundle/decentralized-networks-1d5e2e92953b
2. Erdos P. On Random Graphs. I / P.Erdos, A.Renyi. // Publicationes Mathematicae. – 1959.– №6.– С.290–297.
3. Guillaume J. Bipartite graphs as models of complex networks / J. Guillaume, M. Latapy. // CAAN. – 2004. – С. 215–221.
4. Newman M. Random graphs with arbitrary degree distributions and their applications / M. Newman, S. Strogatz, D. Watts. // Phys. Rev. E. – 2001. – №64.
5. Степеневий розподіл [Електронний ресурс] – Режим доступу до ресурсу: uk.wikipedia.org/wiki/Степеневий_розподіл.
6. Watts D. Networks, Dynamics, and the Small-World Phenomenon / D.J. Watts. // American Journal of Sociology. – 1999. – №105. – С. 493–527.

7. Barabasi A. Emergence of scaling in random networks / A. Barabasi, R. Albert. // Science. – 1999. – №286. – С. 509–512.
8. Анищенко К.М., Жданова Е.Г., Скорик В.А., Сперкач М.О. Исследование возможностей злоупотреблений в децентрализованных сетях с распределением награды // INNOVATIVE SOLUTIONS IN MODERN SCIENCE. – 2019. – №2. – с. 46–62.

УДК 004.021

АКІМОВ Д.Д.
ЖУРАКОВСЬКА О.С.

КОМПЛЕКС ЗАДАЧ ІНФОРМАЦІЙНОЇ ПІДТРИМКИ ПРОЦЕСУ СТВОРЕННЯ МЕДІАМАТЕРІАЛІВ

В статті розглядається комплекс задач, що необхідні для підтримки процесу створення медіа матеріалу у сучасному кінематографі, що базується на детальному аналізі даної галузі. Реалізація процесу підтримки створення медіаматеріалв дозволить спростити цей процес за рахунок ефективного призначення завдань та виконавців а також зменшити ччасові та фінансові витрати на цей процес.

КЛЮЧОВІ СЛОВА: Інформаційна підтримка процесу створення медіаматеріалів, задача про призначення

The article deals with a set of tasks that are necessary to support the process of creating media material in contemporary cinema, based on a detailed analysis of this industry. Implementation of the process of supporting the creation of media materials will simplify this process by effectively assigning tasks and executors, as well as reducing the time and cost of this process.

KEYWORDS: Information support for the process of creation of media materials, the task of appointment

1.Вступ

Сьогодні одна з основних проблем, що негативно впливає на сферу кінематографа, є неефективність використання часу та фінансових ресурсів, і як наслідок, низький рівень продуктивності праці. Це пов'язано зі збільшенням обсягу матеріалу, що потрібно опрацювати для створення якісного медіаматеріалу, що зацікавить глядачів, і зробить проект рентабельним. У зв'язку з цим досить важливо розробити комплекс задач, об'єднаних в систему, що консолідує в собі вирішення усіх проблем, що виникають в процесі створення та аналізу медіаматеріалів.

2.Постановка задачі

Ефективний розподіл задач, та виконавців при створенні медіаматеріалів є найбільш актуальною задачею в цьому процессі.

Користувачами системи являються фахівці кіноіндустрії, починаючи з операторів і закінчуючи режисером та продюсером.

Значний обсяг задач, що вирішуються в процесі роботи над проектом, створює необхідність вирішення задачі про призначення виконавців для множини робіт, мінімізувавши при цьому час на їх виконання. Вказана задача може бути сформульована як класична задача лінійного програмування «Задача про призначення».

Наведемо математичну постановку задачі [1].

Дано n виконавців та m робіт. Будь-який виконавець може бути призначений до виконання будь-якої роботи. Виконання виконавцем роботи пов’язане з витратами, які залежать від того який виконавець виконує роботу. Це відображено коеіцієнтами c_{ij} - вартість виконання працівником і роботи j.

Необхідно виконати всі роботи, призначивши для кожної роботи лише одного виконавця, тобто, знайти таке призначення робіт для виконавців, так, щоб загальні витати були мінімальними. Математична постановка задачі може бути виконана таким чином:оптимізувати задану цільову функцію з врахуванням обмежень:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \to \min \qquad (1)$$

за умов

$$\sum_{j=1}^{m} x_{ij} = 1, \quad i = \overline{1,n} \qquad (2)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = \overline{1,m} \qquad (3)$$

$$x_{ij} \in \{0,1\} \quad i = \overline{1,n},\ j = \overline{1,m} \qquad (4)$$

З урахуванням специфічних особливостей задачі про призначення існує ряд алгоритмів її розв'язання, одним з яких є так званий угорський метод [1].

3.Аналіз досліджень

Акторами системи є режисер, продюсер та виконавець. Для розуміння можливостей користувачів у системі наведемо UML діаграму варіантів використань (Рис.1).

Для визначення структури системи було проведено дослідження існуючих аналогів.

Одним з найпопулярніших у світі конкурентів розроблюваної системи є веб-сервіс для організації як персональної, так і командної роботи Todoist [2]. Дизайн у вигляді календаря, з зображенням поставлених задач на обраний день, який можуть переглядати ті члени команди, для яких призначений запис, є безумовно одною з найвагоміших переваг даного сервісу. Однак, зрозуміло, що значним недоліком сервісу є те, що роботи розподіляються керівником без застосування засобів автоматизації.

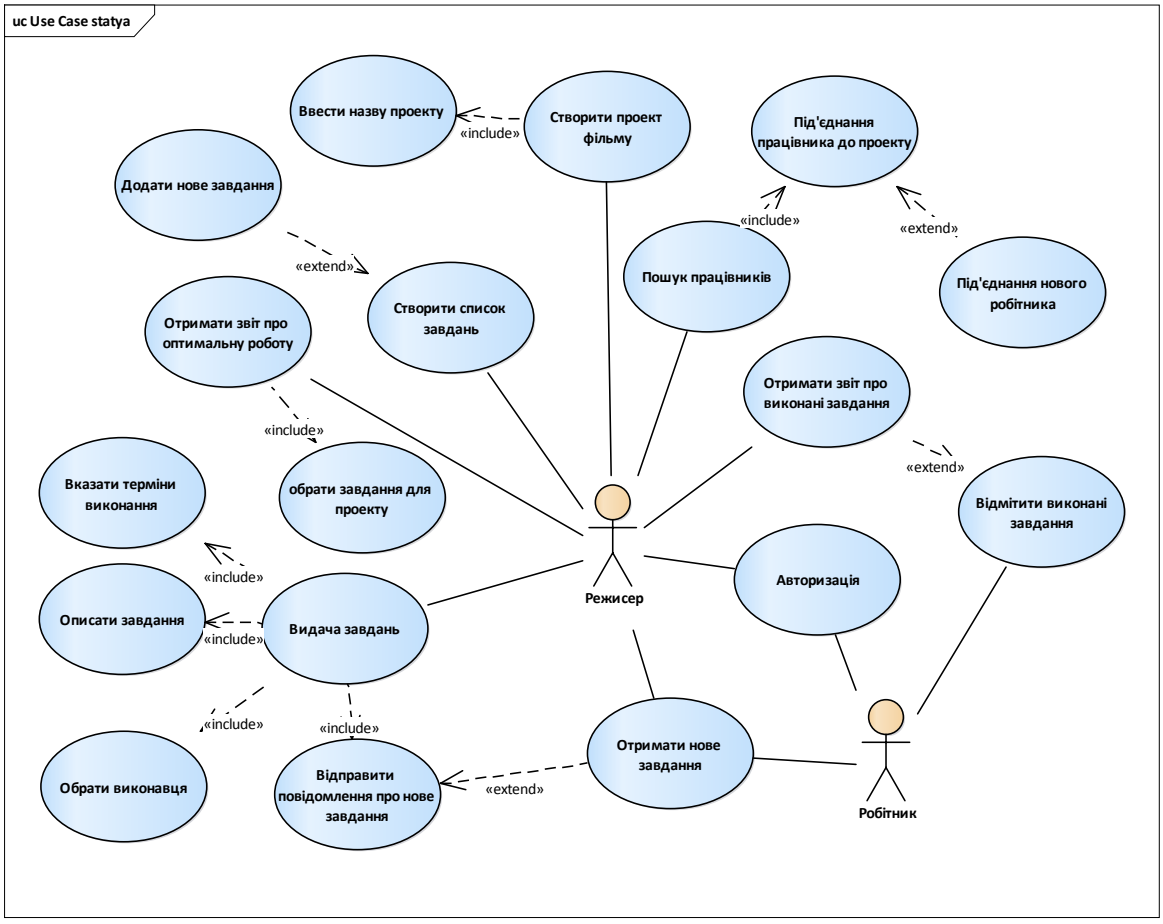


Рисунок 1 – Діаграма варіантів використань

Отже, необхідність створення комплексу задач, та системи, що їх вирішує являється дуже необхідним програмним забезпеченням для працівників кіноіндустрії.

В процесі розробки комплексу задач для інформаційної підтримки процесу створення медіаматеріалу були визначені вхідні та вихідні дані, а також спроектовано базу даних.

В результаті вирішення задачі про призначення, створюється розклад з оптимальним розподілом задач. Після затвердження плану керівником, виконавці отримують план виконання завдань. Для побудованих розподілів формується звіт.

Висновки

В даній статі було обґрунтовано доцільність створення комплексу задач інформаційної підтримки процесу управління створення медіа матеріалу. Були висунуті основні вимоги до системи, виявлені основні групи користувачів, для яких може бути корисною дана система. Було поставлено задачу призначення виконавців множини робіт. Обрано алгоритмічне забезпечення її вирішення. Проведено порівняльний аналіз існуючих засобів - органайзерів для різних платорм. Буда визначена структура системи, описані основні її компоненти, визначена ефективність роботи системи.

Аналіз вирішення поставленої задачі розподілу виконавців та завдань дозволяє зробити висновок про зменшення використаних ресурсів в процесі створення медіаматеріалів.

Список посилань

1. Л.Ф.Гуляницький, О.Ю.Мулеса Методи комбінаторної оптимізації: теоретичні відомості. –Ужгород, 2015.–25 с

2. Офіційний сайт сервісу"Todoist" [Електронний ресурс].- Режим доступу: https://todoist.com/

УДК 004.91

ОЛІЙНИК А.О.

ОГЛЯД ПІДХОДІВ РОЗПІЗНАВАННЯ МАШИНОПИСНИХ ТЕКСТІВ

В даній статті розглядаються підходи розпізнавання машинописних текстів: метод співставлення шаблонів, статистичний метод, структурний підхід та нейронні мережі. Здійснюється порівняння даних методів основуючись на перевагах та недоліках кожного.

Ключові слова: Оптичне розпізнавання символів, співставлення шаблонів, статистичний метод, структурний метод, нейронні мережі, метод опорних векторів, k-найближчих сусідів, багатошаровий перцептрон, порівняння підходів розпізнавання машинописного тексту

In this article, the approaches of optical character recognition, such as template matching, statistical method, structural approach and neural networks, are reviewed. A maked methods comparison based on the advantages and disadvantages each of them.

Keywords: Optical character recognition, OCR, template mapping, statistical approach, syntactic or structural approach, neural networks, support vector machine, SVM, k-nearest neighbors, k-NN, multilayer perceptron, MLP

Вступ

На сучасному етапі розвитку людства задача сканування та зберігання в пам’яті комп’ютера текстових документів є вирішеною. Це надає доступ до них різним користувачам та полегшує роботу довгострокового збереження інформації. Простими та швидкими способами отримання електронної версії документа є сканування та фотографування. Проте, в результаті текст зберігається у вигляді зображення, що є незручно для пошуку, редагування та аналізу.

Для вирішення цієї проблеми необхідно провести ідентифікацію символів. Задача розпізнавання машинописного тексту називається оптичне розпізнавання символів (optical character recognition, OCR). Перша система, що могла виконувати цей процес, була створена в 1970-х роках. З того часу якість роботи розпізнання машинописних текстів поліпшується і зараз на цілком високому рівні. Існує багато програмних високоточних систем, що ідентифікують символи, такі як Abbyy FineReader і Adobe Acrobat Pro DC.

Методи розпізнавання текстів

Існує багато методів розпізнавання тексту. Основною відмінністю є набір характеристик, що ідентифікуються. В даній статті буде розглянуто такі методи OCR: шаблонний метод, статистичний метод, структурний метод та нейронні мережі.

Підхід співставлення шаблонів

Основною ідеєю даного підходу є визначення подібності між об'єктами на зображенні та шаблоном [1]. Степінь схожості знаходиться за допомогою мінімума (максимума) функції співставлення зображення та його прототипу. Цей метод є простим у використанні, так як єдиною складністю стає вибір потрібного шаблону та точної функції. Проблемою використання даного підходу може стати неякісне зображення або будь-яке, навіть незначне відхилення від існуючи шаблонів. Приклад роботи даного підходу представлений на Рис.1. У такому випадку використовують модифікацію методу у вигляді набору логічних правил [2].

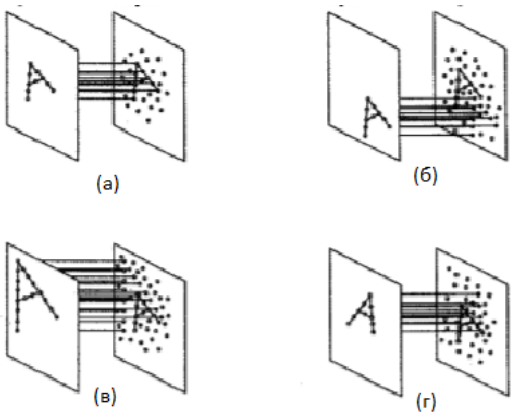


Рис.1. Приклад співставлення шаблону. Подібний шаблону - (а), зміщений – (б), змінений масштаб – (в), повернутий – (г) [3]

Статистичний метод

При використанні даного методу застосовуються статистичні функції прийняття рішень та критерій оптимальності, що визначає максимум ймовірності виникнення закономірності зображення з урахування шаблону певного класу [4]. Широко розповсюджений приклад статистичного методу є метод опорних векторів (support vector machine, SVM). Його задача у тому, щоб визначити, до якого класу шаблонів відноситься дане зображення. Іншим прикладом підходу є правило k-найближчих сусідів (k-nearest neighbors, k-NN). Суть алгоритму у тому, що при надходженні нових даних визначаються k найближчих точок, що вже є маркірованими, і за допомогою них визначається потрібний клас для нових точок.

Структурний підхід

Ідеєю структурного методу є рекурсивний опис складного об'єкту за допомогою простих шаблонів. Найпростіші прототипи називаються примітивами, і саме зображення представлено у вигляді зв'язків між ними. Зазвичай вони являють собою алфавіт, тому зображення текстового документу можна описати за допомогою примітивів і граматики [5]. При такому підході не є важливим розмір або шрифт букви.

Нейронні мережі

З розвитком напряму Data Science нейроні мережі набули великої практичності. Нейромережевий підхід став універсальний для розв'язку задачі розпізнавання. Даний метод описується як масивні паралельно обчислювальні системи, які містять велику кількість нейронів, що взаємопов'язані. Для визначення того, на скільки мережа навчена, використовують функцію втрат. Якщо вона набуває мінімального значення, то якість роботи мережі вважається високою [6]. Прикладом практичного використання методу є багатошаровий перцептрон (multilayer perceptron, MLP). На Рис.2 показано архітектуру даної мережі. Його відмінною ознакою є наявність для кожного нейрону нелінійної функції, існують приховані шари та висока степінь зв'язності.

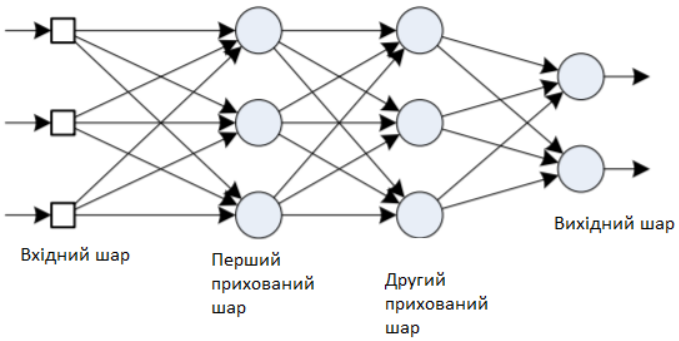


Рис.2. Архітектура багатошарового перцетрону [7]

Ці властивості забезпечують високу потужність, достатню точність та швидкість роботи. З іншої сторони алгоритм є складний через теоретичний аналіз математичної моделі [7]. В результаті багатьох зусиль отримуємо велику точність, але є можливість поліпшення коректності розпізнавальних даних [8]. Іншим прикладом є мережі радіально базисних функцій (radial basic function networks, RBF networks). Відмінністю є прихований шар у вигляді радіальних елементів. На Рис.3 показано архітектуру даної мережі. Перевагами цього алгоритму, порівняно з попереднім, є наявність лише однієї нелінійної функції та оптимізація параметрів лінійної комбінації на вихідному шарі. Отже, мережі радіально базисних функцій працюють швидше ніж багатошаровий перцетрон.

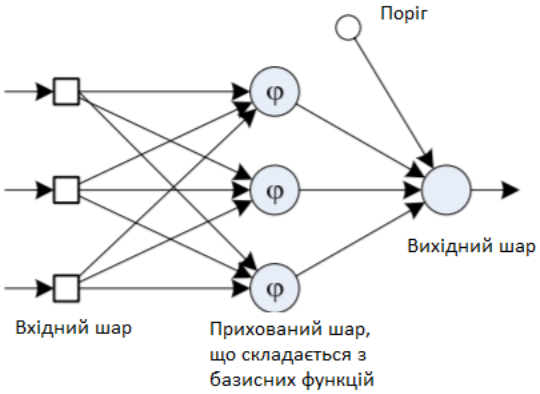


Рис.3. Архітектура мережі радіально базисних функцій [7]

Порівняння методів розпізнавання тексту

Порівняння методів з перевагами та недоліками наведено в Табл.1.

Табл.1. Порівняння методів розпізнавання

Назва підходу	**Переваги**	**Недоліки**	**Програмні продукти**
Шаблонний метод	Простий у використанні, розумінні та реалізації.	Строго залежить від шаблону, при будь-якому відхилені не працює; значна ймовірність наявності помилок; при зміні масштабу, позиції також не працює.	Trapeze компанії SoftWorks AI, Tesseract
Статистичний метод	Ефективність за обмеженої кількості даних для навчання; наявна оцінка результату [6].	Може бути повільним, особливо на великих обсягів даних; не дуже висока точність [6].	Google Web 1T
Структурний підхід	Немає залежності від розміру та шрифту тексту; можна використовувати одну й ту ж граматику.	Складно визначити клас зображення, якщо містить багато примітивів.	Zonal OCR
Нейронні мережі	Забезпечує точність та швидкодію.	Складний з точки зору теоретичної та математичної основи.	Tesseract OCR, Google Doc, ML Kit's text recognition, ABBYY FineReader

Висновок

У ході дослідження проведено аналіз основних підходів розпізнавання машинописних текстів, визначено їх переваги та недоліки. Порівнюючи дані підходи, можна визначити, що найбільшу точність та швидкодію дають нейронні мережі, проте вони є складними в розумінні та налаштуванні. З іншої сторони метод співставлення шаблонів є простим для реалізації, але результат може містити похибки.

Список літератури

1. Jain K., Duin R.P.W., Mao J.: Statistical pattern recognition: A review - IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22 (1) - с.4-8.
2. Бондаренок А.В., Галактионов В.А., Горемычкин В.И., Ермаков А.В., Желтов С.Ю. Исследование подходов к построению систем автоматического считывания символьной информации - ИПМ им. М. В. Келдыша, 2003 – с. 9-13
3. Lecture object recognition [Електронний ресурс]// http://cmp.felk.cvut.cz – 2004 - Режим доступу до ресурсу: http://cmp.felk.cvut.cz/cmp/courses/dzo/resources/lecture_object-recognition.pdf
4. Goswami Richa, Sharma O.P. A review character recognition techniques – International Journal of Computer Applications (0975-8887) - №7, 2013 – с.21-22
5. Dr.B.Rama, Santosh Kumar Henge OCR-The2 layered approach for classification and identification of telugu hand written mixed consonants and conjunct consonants by using advanced fuzzy logic controller // Department of Computer Science, Kakatiya University, Warangal,India – 2016 – с.77-78
6. Лавренюк М.С., Новіков О.М. Огляд методів машинного навчання для класифікації великих обсягів супутникових даних - System Research & Information Technologies, 2018, № 1. - С. 54-57
7. Воронцов И.В., Политов Е.А., Ефременко В.М. Обзор типов искусственных нейронных сетей и методов их обучения // Вестник Кузбасского государственного технического университета, 2007 – с.39-40
8. Singh Vijendra, Nisha Vasudeva, Hem Jyotsana Parashar Recognition of Text Image Using Multilayer Perceptron [Електронний ресурс]// https://arxiv.org – 2016 - Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1612/1612.00625.pdf

УДК 004.93

МАДОЯН Г.О.,
ШИШМАН Ю.М.

ІНФОРМАЦІЙНА СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ КОРИСТУВАЧЕМ ПІД ЧАС ПРОЦЕСУ СОРТУВАННЯ ВТОРИННОЇ СИРОВИН

В статті розглянуто впровадження інформаційної ситеми підтримки утилізації відходів з метою збільшити обізнаність користувачів у процесі сортування вторинної сировини та полегшити процес сортування. Реалізовано розпізнавання зображень та вбудовано чат-бот для отримання релевантних відповідей стосовно сортування та пунктів прийому. Приведено аналіз ефективності чат-боту та алгоритму розпізнавання зображень, визначено основні показники якості – точність, прискорення, ефективність, актуальність. Наведено графіки залежності точності розпізнавання від кількості процесорів та графіки даних NLU.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, ВТОРИННА СИРОВИНА, АЛГОРИТМ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, ПРИСКОРЕННЯ, ЕФЕКТИВНІСТЬ, ДЕРЕВО РІШЕНЬ, ЧАТ-БОТ.

The article review the implementation of a waste management information system in order to increase user awareness in the process of sorting secondary raw materials and to facilitate the sorting process. Image recognition is implemented and the chat bot is built-in to provide relevant responses regarding sorting and receiving points. An analysis of the effectiveness of the chat-bot and image recognition algorithm are given, the main indicators of quality are defined - accuracy, acceleration, efficiency, relevance. Graphs of dependence of recognition accuracy on the number of processors and data graphs of NLU are presented.

KEYWORDS: INFORMATION SYSTEM, IMAGE RECOGNITION ALGORITHM, PARALLEL COMPUTING, ACCELERATION, EFFICIENCY, DECISION TREE, CHATBOT.

1. Вступ

Проблема сміття в Україні стоїть особливо гостро - 95% всіх твердих відходів потрапляють на звалища. Кожен українець щорічно генерує близько 350 кг сміття. А в рік на полігони і неофіційні звалища вивозять приблизно 14 мільйонів тон побутових відходів. Загальна площа всіх звалищ складає майже 5% території країни [1].

В Україні процес сортування сміття стає більш популярним, люди розуміють шкоду, яку чинять навколишньому середовищу і врешті решт починають сортувати вторинну сировину. Але при цьому стикаються з рядом труднощів:

- відсутність розуміння категорій відходів;
- відсутність розуміння конкретної сировини, до якого типу відходів може відноситися;
- брак інформації щодо пунктів прийому сировини.

З огляду на це є актуальним питання розробки інформаційної системи підтримки утилізації відходів (ІСПУВ), яка б водночас змогла б підтримувати розпізнавання зображень вторинної сировини та надавала рекомендації щодо попередньої обробки та доставки вторинної сировини в пункти збору.

2. Постановка задачі

Призначенням ІСПУВ є підтримка прийняття рішень під час процесу сортування вторинної сировини за категоріями (пластик, деревина, скло, папір, метал, одяг).

Метою розробки інформаційної системи є:

- збільшення обізнаності користувачів у процесі сортування вторинної сировини;
- полегшення процесу сортування вторинної сировини;
- отримання релевантних відповідей на запити користувачів стосовно сортування вторинної сировини та пунктів її прийому за прийнятний час.

Важливими частинами ІСПУВ є: підсистема розпізнавання зображень та підсистема комунікації з користувачем за допомогою чат-боту.

3. Чат-бот

Бот – це комп'ютерна програма, призначена для людей, яка автоматично і / або за деякою послідовністю виконує певні дії через текстовий чат. Dialogflow – це онлайн сервіс від компанії Google, який дозволяє створювати чат-ботів різного призначення. Сервіс дозволяє створювати розмовні інтерфейси поверх програмних продуктів і послуг, забезпечуючи потужний механізм розуміння природної мови (NLU) для обробки та розуміння текстів природними мовами.

Для розробки чат-ботів Dialogflow надає агентів, сутності та наміри. Агент (Agent) - це і є чат-бот, який може зрозуміти великі і різноманітні нюанси людської мови. Намір (Intent) - це зіставлення між тим, що говорить користувач, і які операції може виконати бот. Сутність (Entity) - це група об'єктів, які необхідно розпізнати агенту.

Агенти Dialogflow використовують алгоритми машинного навчання, щоб зрозуміти висловлювання природною мовою, зіставити їх з намірами і витягти структуровані дані.

Агент навчається як від фраз навчання, які ви надаєте, так і від мовних моделей, вбудованих у Dialogflow. Виходячи з цих даних, він будує алгоритм для прийняття рішень про те, який намір повинен бути узгоджений з висловом користувача. Цей алгоритм є унікальним для вашого агента.

Dialogflow оновлює алгоритм машинного навчання вашого агента щоразу, коли ви вносите зміни до намірів і об'єктів, імпортуєте або відновлюєте агента або тренуєте свого агента. [2].

На рисунку 1 зображено частину детермінованого дерева рішень (підкатегорії пластику), яке відображає можливі варіанти рішень користувача та отриману користь при виборі того чи іншого стану при роботі з чат-ботом. Процес прийняття рішень за допомогою дерева рішень виконується в п'ять етапів:

Етап 1. Формулювання завдання. Насамперед необхідно відкинути всі фактори, що не стосуються проблеми, а серед множини тих, що залишилися, виділити істотні та несуттєві.

Етап 2. Побудова дерева рішень.

Етап 3. Оцінка імовірностей станів середовища, тобто зіставлення шансів виникнення кожної конкретної події.

Етап 4. Встановлення виграшів (чи програшів, як виграшів зі знаком мінус) для кожної можливої комбінації альтернатив (дій) і станів середовища.

Етап 5. Вирішення завдання. «Дерево рішень» складається з ряду вузлів і гілок, які з них виходять [3]. Квадрати позначають пункти прийняття рішень, кола – можливі події, а дуги − відповідають переходам між логічно пов'язаними рішеннями і випадковими подіями. З вершин-рішень (квадратів) виходить стільки дуг, скільки є варіантів (альтернатив); вибір конкретної дуги (варіант рішення). Для розробленої підсистеми надання рекомендацій щодо попередньої обробки та доставки вторинної сировини в пункти збору дерево рішень є детермінованим, оскільки користувач в будь-якому випадку отримує користь, навіть якщо він не знайшов точної відповіді на своє питання, то чат-бот надасть інформацію про те, де можна знайти відповідь або до кого звернутися.

Для аналізу ефективності розробленого продукту можемо переглянути статистику, що стосується конкретного агенту. Він допомагає обробляти вхідні дані від користувача в структуровані дані, які можна використовувати для повернення певної відповіді. На рисунку 2 відображено графік сесій, коли користувач взаємодіє з чат-ботом. Значення 56 над графіком означає кількість сеансів за попередній день. Синя лінія на графіку показує дані за поточний день або період часу, а вторинна пунктирна лінія (світло-блакитна) позначає дані за попередній день або період часу, тому ми можемо порівняти останні зміни.

На рисунку 3 представлено приклад статистичних даних про розмовні шляхи, які користувачі системи обрали при взаємодії з агентом, а також відсоток всіх користувачів, які отримали відповідь на поставлене запитання та кількість запитів, яким було співставлено намір.

4. Алгоритм розпізнавання зображень

Значним класом задач, які вирішуються за допомогою нейронних мереж є задачі класифікації. Одна з найдавніших задач, що стоїть перед мережею: розпізнавання об'єкта на зображенні. Передбачається, що правильна відповідь - строго одна, а все інше, що потрапило в кадр - шум.

При навчанні з учителем нейронна мережа навчається на розміченому наборі даних і передбачає відповіді, які використовуються для оцінки точності алгоритму на навчальних даних. При навчанні без учителя модель використовує нерозмічені дані, з яких алгоритм самостійно намагається витягти ознаки і залежності. Модель навчання з учителем складається з трьох взаємопов'язаних компонентів, які в математичних термінах описуються наступним чином:

1) Середовище: характеризується розподілом ймовірностей $P_X(X)$ з випадковими і незалежними величинами X, які з'являються.

2) Вчитель: генерує бажаний відгук d для кожного з вхідних векторів X, отриманих із зовнішнього середовища, відповідно до умовної функції розподілу $P_X(d|X)$. Ні характеристика середовища $P_X(X)$, ні правило класифікації $P_X(d|X)$ невідомі. Однак відомо, що обидві функції існують, тобто існує спільний розподіл ймовірностей:

$$P_X(d|X) = P(x) * P_X(d|X),$$

Бажаний відгук d та вхідний вектор X зв'язані наступним співвідношенням:

$$d = f(X, v),$$

де v – шум, тобто зарані передбачається зашумленість даних вчителя.

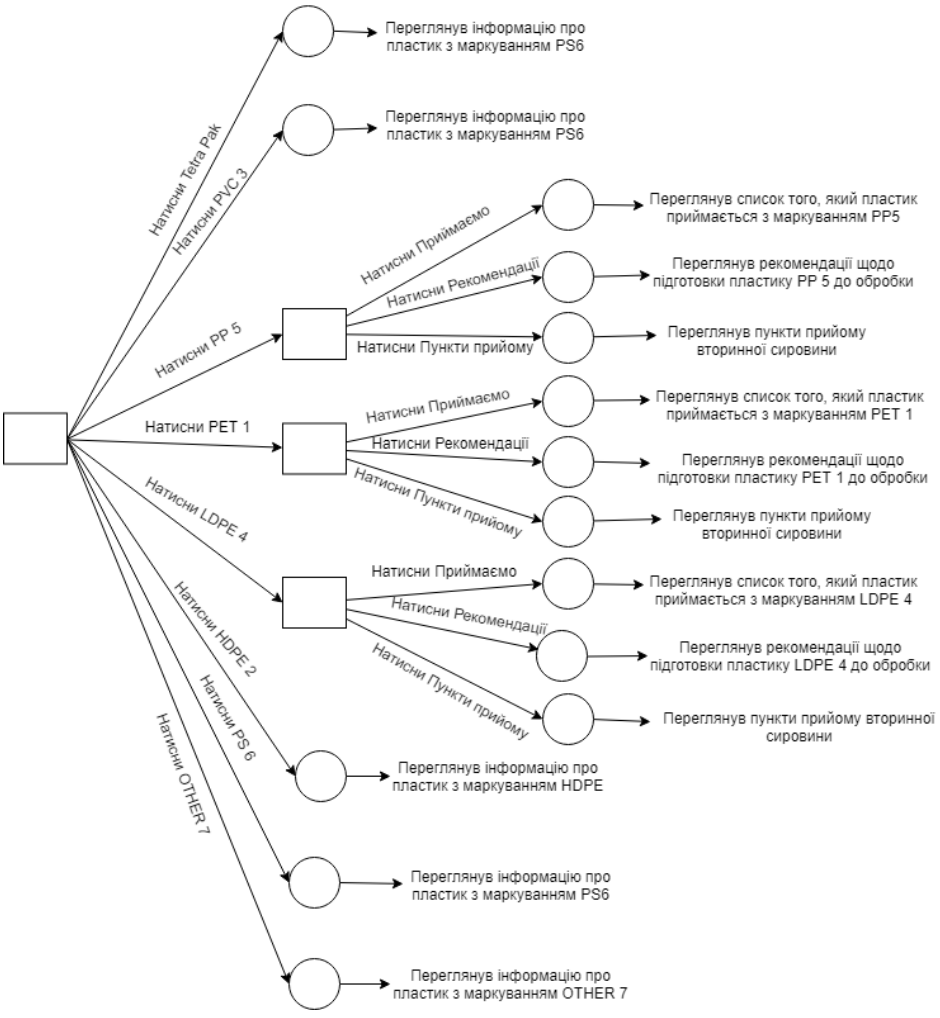


Рис. 1. Детерміноване дерево рішень (підкатегорії пластику)

3) Навчальна машина: нейронна мережа здатна реалізувати функції відображення вхід-вихід, описуваних співвідношенням

$$y = F(x, w),$$

де y - фактичний відгук, згенерований навченою машиною у відповідь на вхідний сигнал x ; w - набір вільних параметрів (синаптичних ваг), вибраних з простору параметрів W.

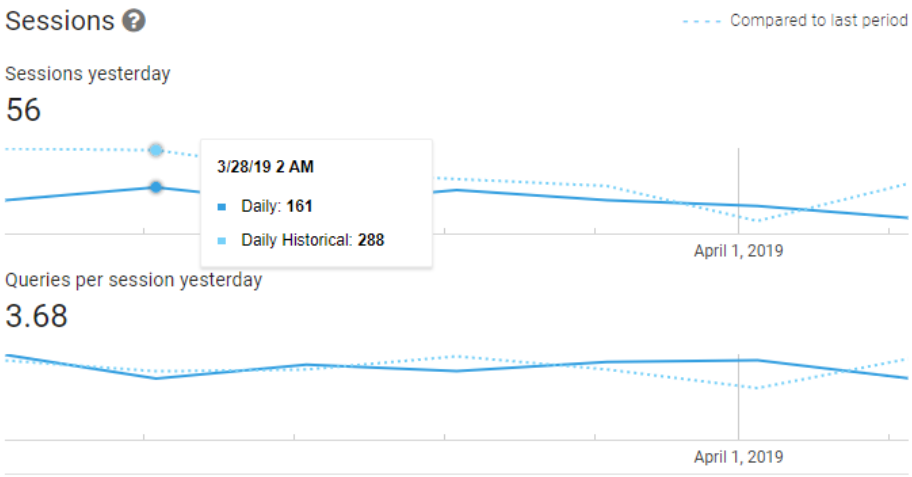


Рис. 2. Статистичні дані денного сеансу

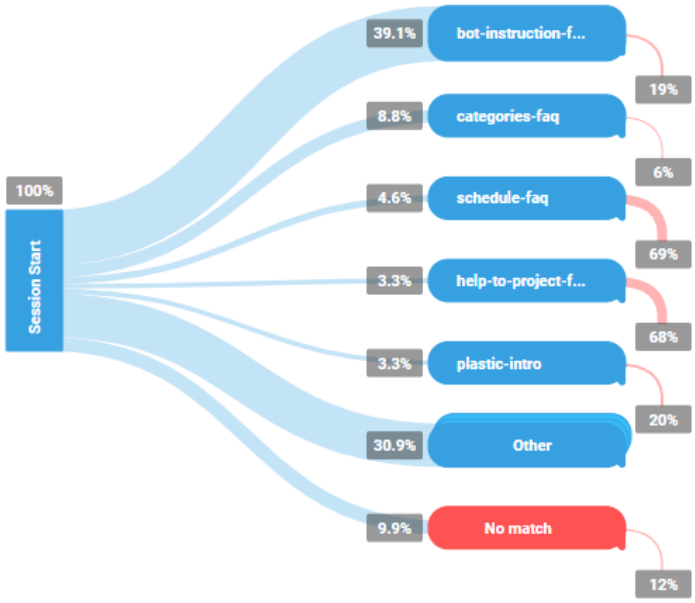


Рис. 3. Діаграма потоку сеансу

Задача навчання з учителем полягає у виборі конкретної функції $F(x, w)$, яка оптимально (в деякому статистичному сенсі) апроксимує очікуваний відгук d. Вибір, в свою чергу, ґрунтується на множині N незалежних, рівномірно розподілених прикладів навчання:

$$T = \{(d_i, x_i)\}, i = 1, \dots, N.$$

Найбільш поширеним алгоритмом навчання нейронної мережі є метод зворотного поширення помилки, який є модифікацією методу градієнтного спуску. На основі обраної функції вихідної помилки $E(y_i, \hat{y}_i)$, де y – цільове значення виходу, $\hat{y}$ – поточне вихідне значення. Від вибору функції оцінки залежить ефективність навчання. На кожній ітерації методу виконується коригування ваг кожного шару нейронної мережі за наступною схемою:

$$W_i^{(l)} = W_i^{(l)} + \eta \Delta W_i^{(l)},$$

де l – номер шару, i – номер ваги на поточному шарі, η – параметр швидкості навчання.

Величина, на яку проводиться коригування ваг, визначається так:

$$\Delta W_i^{(l)} = -\delta_i^{(l)} \frac{\partial a(Z_i)}{\partial Z_i} y_i^{(l-1)},$$

де $a(Z)$ – активаційна функція поточного шару, $Z_i = \sum_{j=1}^{N} w_j^{(l)} y_j^{(l-1)}$ - зважена сума значень виходів попереднього шару $y_j^{(l-1)}$, N – кількість нейронів у мережі, $\delta_i^{(l)}$ – значення параметру похибки.

Параметр похибки визначається наступним чином:

$$\delta_i^{(l)} = \frac{\partial E(y_i, \hat{y_i})}{\partial \hat{y_i}}$$

(для вихідного шару)

$$\delta_i^{(l)} = \sum_{j=1}^{N} \delta_j^{(l+1)} w_j^{(l)}$$

(для прихованих шарів)

Для визначення якості роботи нейронної мережі використовують функцію втрат (loss function). Зазвичай за таку функцію обирають евклідову відстань, середньоквадратичну похибку або функцію кросентропії. Мережа вважається навченою, якщо функція втрат набуває мінімального значення. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів у напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи мережі. Процедуру зворотного поширення помилки можна застосувати кілька разів, щоб поширити градієнти через усі шари, починаючи з виходу (результату прямого проходження нейронної мережі) і до входів, що подаються в мережу. У процесі навчання нейронної мережі ваги зв’язків між нейронами коригуються на основі методу градієнтного спуску (gradient descent). На практиці зазвичай використовують модифікацію цього методу, коли процедура градієнтного спуску застосовується до груп навчальних прикладів. Такий підхід називається методом стохастичного градієнта (stochastic gradient descent) [4], що значно пришвидшує час навчання нейронної мережі.

Схема алгоритму

Крок 1. Вагам мережі присвоюються невеликі початкові значення.

Крок 2. Вибирається чергова навчальна пара *(X, Y)*; вектор *X* подається на вхід мережі.

Крок 3. Обчислюється вихід мережі.

Крок 4. Обчислюється різниця між виходом мережі, що вимагається (цільовим, *Y*) і реальним (обчисленим).

Крок 5. Вага мережі корегується так, щоб мінімізувати помилку.

Крок 6. Кроки з 2-го по 5-й повторюються для кожної пари навчальної множини до тих пір, поки помилка на всій множині не досягне прийнятної величини.

Метод зворотного поширення помилки можливо реалізувати в двох режимах:

– стохастичний градієнтний спуск - після обчислення виходу вводяться поправки в ваги. Повільний, але здатний виходити з локальних екстремумів;

– пакетний градієнтний спуск - після проходження епохи вносяться поправки в ваги. Швидкий, але висока ймовірність зупинитися в локальному екстремумі.

Для коригування вагів реалізуємо модифікацію градієнтного спуску, що є його оптимізацією - метод стохастичного градієнтного спуску, коли на кожній ітерації алгоритму з навчальної вибірки випадковим чином обирається лише один об'єкт. Таким чином вектор ваг налаштовується кожен раз на новобраний об'єкт. Використовується для прискорення пошуку цільової функції шляхом використання обмеженого за розміром тренувального набору, який вибирається випадкового при кожній ітерації.

Для зменшення часу обчислень алгоритму підвищення швидкодії деякі кроки можуть бути розділені на паралельні задачі і виконуватися паралельно. Паралельна обробка є об'єднанням декількох процесорів для вирішення завдання. Це дозволяє вирішувати об'ємні завдання і завдання фіксованої розмірності набагато швидше.

Під час навчання нейронних мереж є потреба в збільшенні швидкості обробки даних за рахунок разпаралелювання алгоритму.

5. Дослідження ефективності розпаралелювання алгоритму

Найбільш загальною формою подання алгоритмів є інформаційно-керуючий граф алгоритму, який відображає залежність за даними між операторами алгоритму і безумовні і умовні переходи в програмі.

Більш певною формою подання паралелізму задач є апарат ярусно-паралельної форми. Ярусно-паралельна форма - це такий вид графа, у якого в верхній нульовий ярус поміщені вершини, що мають тільки вихідні дуги; в нижній ярус поміщені вершини, що мають лише вхідні дуги. На рисунку 4 зображена ярусно-паралельна форма алгоритму.

На графі позначені переходи, які означають передачу результатів обчислення примітивної операції з одного ярусу до операції з наступного ярусу. Це можна назвати викликами методів в алгоритмі, в які передаються певні дані, які були результатами методів, які виконувались перед цим. Яруси діляться по переходах.

Розпаралелимо частину алгоритму, яку представляє собою метод стохастичного градієнтного спуску. Ми маємо набір тренувальних даних і для них розраховуються ваги. Ці розрахунки є незалежними, тобто обчислення ваг можна виконувати паралельно: одна частина даних оброблюється один процесом, друга – другим і т.п. За рахунок цього можна досягти зменшення часу обчислення.

В даному випадку рішення проблеми представляється у вигляді колекції одночасно виконуваних однакових операцій, саме тому розділення на паралельні підзадачі тут доцільне.

Для оцінки розроблюваних методів паралельних обчислень наведені широко використовувані в теорії і практиці паралельного програмування основні показники якості – прискорення (speedup), що показує, у скільки разів швидше здійснюється рішення завдань при використанні декількох процесорів, і ефективність (efficiency), яка характеризує частку часу реального використання процесорів обчислювальної системи.

Можемо вважати, що для вирішення задачі послідовно потрібно часу $T_1 = fT_1 + (1-f)T_1$, а для вирішення паралельно $T_p = fT_1 + \frac{(1-f)T_1}{p}$ [5]. Обчислимо прискорення:

$$S_p = \frac{T_1}{T_p} = \frac{fT_1 + (1-f)T_1}{fT_1 + \frac{(1-f)T_1}{p}}$$

Тепер обрахуємо прискорення роботи програми за формулою Амдала [6]:

$$S_p = \frac{1}{f + \frac{1-f}{p}}.$$

Враховуючи те, що всі операції, що виконується в рамках алгоритму, розпаралелені, частка коду (f), яка не може бути розпаралелена дорівнює 0, враховуючи те, що кількість процесорів дорівнює 4, то p=4. Прискорення оцінюється величиною:

$$S_p = \frac{1}{f + \frac{1-f}{p}} = \frac{1}{0 + \frac{1-0}{4}} = 4.$$

Ефективність використання процесорів (p=4) при паралельній реалізації алгоритму оцінюється величиною:

$$E_p(n) = \frac{T_1}{pT_n} = \frac{S_p}{p} = \frac{p}{p} = 1.$$

Отже, теоретично, розбиття задачі на підзадачі, вирішувані за алгоритмом розпізнавання зображень, є доцільним, адже завдяки цьому досягається прискорення його роботи.

На рисунку 5 зображено залежність прискорення алгоритму від кількості даних на максимальній кількості процесорів, на рисунку 6 – залежність точності розпізнавання від кількості даних, де кількість даних – це число зображень, які подаються на вхід для навчання.

З рисунків видно, що точність навчання значно залежить від кількості даних, що подавалися на вхід. Прискорення зі збільшенням кількості процесорів зростає, проте значення дещо менші за теоретичні.

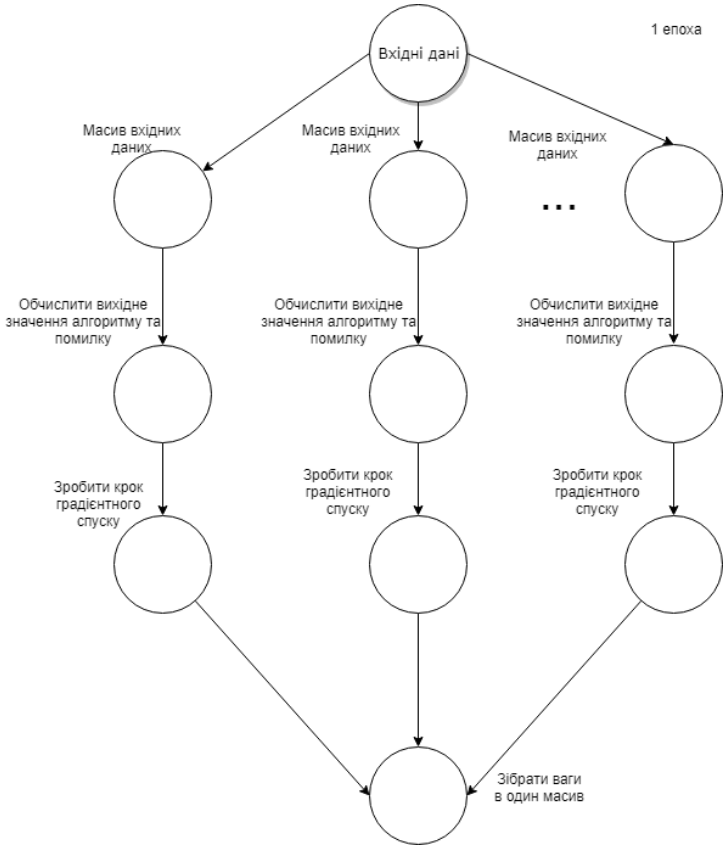


Рис. 4. Ярусно-паралельна форма алгоритму

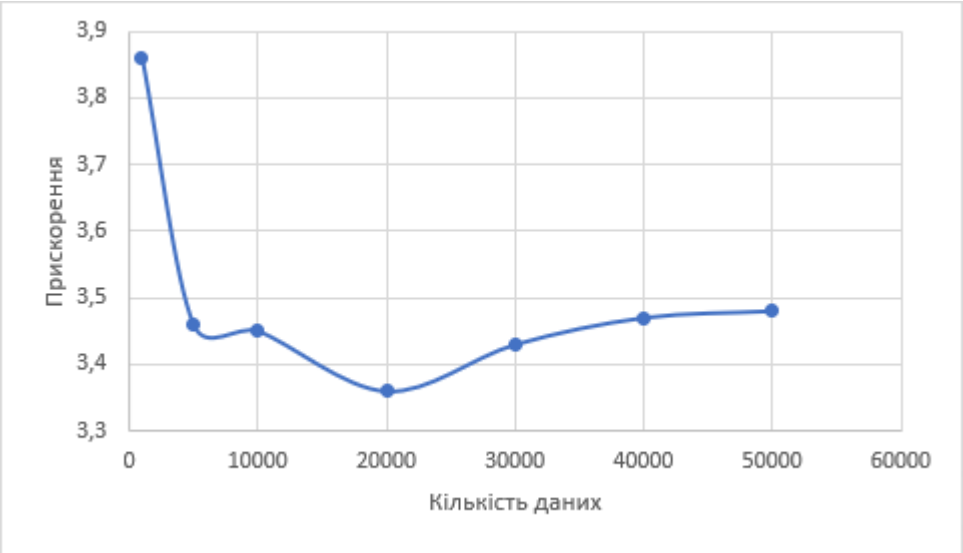


Рис. 5. Залежність прискорення від кількості процесів

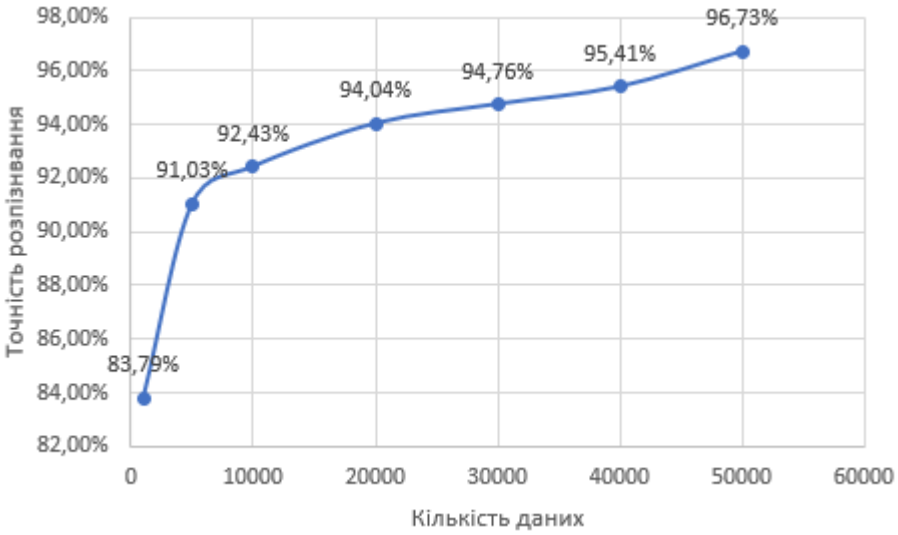


Рис. 6. Залежність ефективності від кількості процесів

Висновки

Впровадження інформаційної системи підтримки утилізації відходів надасть можливість збільшити обізнаність користувачів у процесі сортування вторинної сировини, полегшити процес сортування та отримати релевантні відповіді стосовно сортування та пунктів обробки. Вбудований в систему чат-бот допомагає отримувати релевантні відповіді на запити користувачів стосовно сортування вторинної сировини та пунктів її прийому, дозволяє збільшити обізнаність користувачів у процесі сортування, підсистема розпізнавання зображень допомагає полегшити процес сортування вторинної сировини за рахунок використання нейронної мережі та паралельного виконання алгоритму.

Список використаних джерел

1. Сортування сміття – красиве і корисне [Електронний ресурс]. Режим доступу: https://lb.ua/society/2018/05/02/396399_sortuvannya_smittya_krasive_i_korisne.html
2. Machine learning [Електроний ресурс]. Режим доступу: https://dialogflow.com/docs/agents/machine-learning
3. Використання "дерева рішень" [Електронний ресурс]. Режим доступу: https://pidruchniki.com/10780621/ekonomika/vikoristannya_dereva_rishen
4. Машинне Навчання: Класифікація тексту з Нейронними Мережами та TensorFlow [Електронний ресурс]. Режим доступу: https://codeguida.com/post/818
5. Эффективность параллельных программ [Електронний ресурс]. Режим доступу: http://rsusu1.rnd.runnet.ru/tutor/method/m1/page09_3.html
6. Закон Амдала [Електронний ресурс]. Режим доступу: https://medium.com/german-gorelkin/amdahls-law-79a8edb040e2

УДК 004.023

ІВАНОВА Л.А.,
ОЛІЙНИК Ю.О.

ВИЯВЛЕННЯ ОБ'ЄКТІВ У ПРОСТОРІ ЗА ДОПОМОГОЮ ГЛИБИННОГО НАВЧАННЯ

У цій статті розглянуто задачу виявлення багатьох об'єктів у просторі і алгоритми вирішення даної задачі із застосуванням підходів глибинного навчання. Сформульовано цілі та актуальність дослідження. Виділено загальні задачі розпізнавання образів, розглянуто різницю між задачами. Описано сучасні підходи до вирішення проблеми розпізнавання декількох об'єктів за допомогою алгоритмів глибинного навчання, виявлено їх особливості.

КЛЮЧОВІ СЛОВА: РОЗПІЗНАВАННЯ ОБРАЗІВ, ВИЯВЛЕННЯ ОБ'ЄКТІВ, ГЛИБИННЕ НАВЧАННЯ, КОМП'ЮТЕРНЕ БАЧЕННЯ, НЕЙРОННІ МЕРЕЖІ

This article is concerned with the task of multiple object detection and deep learning algorithms usage for solving this problem. The goals and relevance of the research are formulated. The main tasks of pattern recognition are described, the difference between them is highlighted. The modern approaches to the multiple object detection using deep learning algorithms are described, the features of each approach are provided.

KEYWORDS: PATTERN RECOGNITION, OBJECT DETECTION, DEEP LEARNING, COMPUTER VISION, NEURAL NETWORKS

1. Вступ

Комп'ютерний зір – це технологія створення машин, які можуть проводити виявлення, стеження та класифікацію об'єктів. Розпізнавання образів є галуззю, яка використовує різноманітні методи для отримання інформації з відеоданих. Значна частина цієї області присвячена практичному застосуванню цих методів. [1]

Теорія розпізнавання образів – це розділ кібернетики, що розвиває теоретичні основи і методи класифікації і ідентифікації предметів, явищ, процесів, сигналів, ситуацій та інших об'єктів, які характеризуються кінцевим набором деяких властивостей і ознак.

Створення штучних систем розпізнавання образів залишається складною теоретичною і технічною проблемою. Необхідність у такому розпізнавання виникає у дуже різних галузях промисловості: від військової справи і систем безпеки до медицини. [2]

Поставлена задача: дослідити існуючі алгоритми розпізнавання об'єктів у просторі. Задача поділяється на насупні підзадачі:

1. Дослідження загальних методів розпізнавання образів
2. Дослідження підтипів задач розпізнавання образів
3. Дослідження алгоритмів глибинного навчання у застосуванні до розпізнавання образів.

2. Загальні методи розпізнавання образів

Для оптичного розпізнавання образів можна застосувати метод перебору вигляду об'єкта під різними кутами, масштабами, зсувами тощо. Для букв потрібно перебирати шрифт, властивості шрифту та інші ознаки.

Також використовується підхід знаходження контуру об'єкта і подальшого досліджування його властивостей (зв'язність, наявність кутів тощо)

Ще один підхід — використовувати штучні нейронні мережі. Цей метод вимагає або великої кількості прикладів задачі розпізнавання (із правильними відповідями), або спеціальної структури нейронної мережі, що враховує специфіку даної задачі.

3. Задачі розпізнавання образів

Підтипи задач розпізнавання образів загалом відрізняються за наступними критеріями:

1. Чи потрібно розпізнати один об’єкт або декілька?
2. Чи потрібно знайти розташування об’єкту на зображенні?
3. Якщо потрібно знайти розташування об’єкту, чи потрібно знайти його контури?

Серед задач розпізнавання одного об’єкту можна виокремити задачі класифікації зображення і локалізації об’єкту.

Задача класифікації зображення (image classification) – це задача розпізнавання одного образу на зображенні. У рамках цієї задачі дослідник має зображення і його метою є надати йому одну з багатьох категорій. Питанням, яке вирішує вказана задача, є “Що на зображенні?”.

Локалізація об’єкту (object localization) – це задача розпізнавання одного образу на зображенні і додаткове знаходження розташування розпізнаного об’єкту на зображенні. Для виконання цієї задачі необхідна попередня класифікація об’єкту. Питанням, яке вирішує вказана задача, є “Що на зображенні і де саме воно знаходиться?”. Результатом локалізації зображення зазвичай стають наступні параметри: назва класу об’єкта, координати лівого верхнього кутка рамки об’єкту за осями X і Y, ширина і висота знайденого об’єкта.

Задачі розпізнавання багатьох об’єктів більш часто зустрічаються у практичному використання комп’ютерного зору. Серед задач розпізнавання багатьох об’єктів можна виокремити задачі виявлення об’єктів і сегментацію сутностей.

Виявлення об’єктів (object detection) – це задача локалізації декількох об’єктів різних типів. У рамках цієї задачі дослідник зацікавлений у знаходженні всіх об’єктів різних категорій на зображенні і розташуванні так званих рамок біля знайдених об’єктів. Прикладом вирішення такої задачі є самостійно керований автомобіль: у відео-потоці потрібно знаходити інші автомобілі, світлофори, знаки, людей тощо і маючи цю інформацію приймати рішення.

Сегментація сутностей (instance segmentation) – це завдання подібне до виявлення об’єктів із додатковою вимогою знайти точні контури об’єктів, що розпізнаються на зображенні.

Приклади результату розпізнавання у рамках описаних задач наведено на рисунку 1. [3]

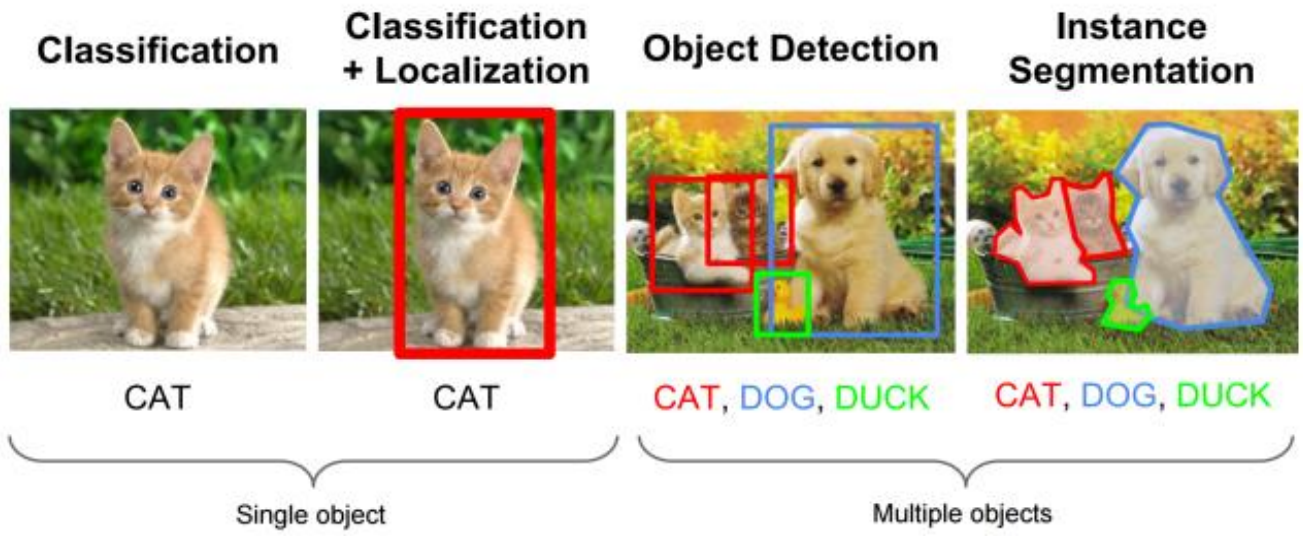


Рис.1. Результати розпізнавання образів

4. Проблеми виявлення об’єктів

Виявлення об’єктів можна вважати проблемою класифікації, коли дослідник використовує вікна фіксованих розмірів, отриманих з початкового зображення на усіх можливих локаціях, і надає їх до класифікатора зображень. Класифікатор визначає клас об’єкту у вікні, і таким чином відомо знаходження об’єкту на зображенні і клас об’єкту.

При використанні такого методу виникає проблема розміру вікна: яким чином визначити розмір вікна, щоб вікно завжди містило об’єкт? Для вирішення цієї проблеми використовують масштабування зображення. Будується “піраміда” зображень за допомогою пропорційного поступового зменшення початкових розмірів. Ідея такого рішення полягає у тому, що якщо поступово зменшувати масштаб оригінального зображення, обране дослідником вікно фіксованого розміру обов’язково буде містити повністю об’єкт з масштабованого зображення.

Існує також проблема співвідношення сторін. Велика кількість об’єктів може бути представлена у різних формах. Наприклад, людина, яка сидить та стоїть, буде мати різні співвідношення сторін.

Є методи, які вирішують задачу виявлення об’єктів і описані проблеми.

5. Використання гістограм орієнтованих градієнтів (HOG)

Гістограми орієнтованих градієнтів було представлено ще у 2005 році Навнітом Далалом і Білом Тригсом. Ці характеристики не є обчислювально-затратними. У кожному вікні обчислюються HOG-характеристики і подаються на вхід до машини опорних векторів (Support Vector Machine) для створення класифікаторів. Такий метод підходить для виявлення пішоходів, розпізнавання обличь та інших сценаріїв виявлення об’єктів.

6. Використання згорткових нейронних мереж на основі регіону (R-CNN)

Якщо звести проблему виявлення об’єктів до проблеми класифікації, успішність рішення залежить від точності класифікації. З появою глибинного навчання з’явилась задача заміни HOG-класифікаторів на більш точні згорткові нейронні мережі (Convolutional Neural Networks). Цьому заважала повільність і обчислювальна затратність CNN. Проблему було вирішено використанням згорткової нейронної мережі на основі регіону (Region-based CNN, скорочено R-CNN). R-CNN використовує алгоритм вибіркового пошуку (Selective Search), який зменшує кількість рамок, що поступають до класифікатора. Вибірковий пошук використовує локальні тригери, такі як текстура, інтенсивність, колір тощо для генерації можливих локацій об’єкта. Таким чином, можна використовувати рамки для класифікатора на основі CNN. Оскільки CNN приймає на вхід дані фіксованого розміру, потрібно змінити без зберігання співвідношення розмір згенерованих рамок. Таким чином, R-CNN містить 3 важливих кроки застосування:

1. Здійснити вибірковий пошук для генерації можливих об’єктів.
2. Подати отримані дані на вхід до CNN, після чого використати SVM для визначення можливого класу кожного об’єкту.
3. Оптимізувати дані за допомогою регресії рамок об’єктів.

Незважаючи на оптимізацію, R-CNN все ще були дуже повільними через використання CNN для багатьох згенерованих об’єктів.

7. Алгоритм SPP-net

З використанням SPP-net дослідник обчислює представлення CNN лише для одного зображення і результат можна використовувати для обчислення представлення CNN кожного можливого об’єкту, згенерованого вибірковим пошуком. Це може бути зроблено шляхом виконання об’єднання лише тих секцій мап властивостей, що відповідають регіону із об’єктом. Прямокутний переріз згорткового шару, який відповідає регіону, можна обчислити за допомогою проекції регіону на згортковий шар із урахуванням зменшення, яке відбувається у проміжних шарах.

Великим недоліком SPP-net було те, що здійснювати зворотнє поширення (back propagation) через об’єднуюичий шар було нетривіальною задачею.

7. Швидкі R-CNN

Цей метод використовує ідеї SPP-net і R-CNN і вирішує ключову проблему SPP-net: із використанням швидких R-CNN можливо здійснювати тренування у форматі end-to-end, включаючи зворотнє поширення. Щоб поширювати градієнти через просторове об’єднання, метод використовує просте обчислення зворотнього поширювання, дуже схоже на розрахунок градієнта з

максимальним об’єднанням, за винятком того, що області об’єднання поєднуються між собою, і тому клітина може мати градієнти з декількох областей.

Додатково, у швидких R-CNN регресія рамок додається до тренування нейронної мережі. Таким чином, не потрябно тренувати нейронну мережу окремо для задач класифікації і локалізації.

8. Швидші R-CNN

Це вдосконалений метод швидких R-CNN. Більшої швидкості у ньому було досягнуто за допомогою заміни вибіркового пошуку маленькою згортковою мережею пропозицій регіону (Region Proposal Network).

Для вирішення проблеми масштабування і співвідношення сторін об’єктів, швидші R-CNN пропонують ідею рамок-якорів. На кожній локації використовуються 3 види рамок 128x128, 256x256, 512x512. Так само, використовуються співвідношення сторін 1:1, 1:2, 2:1. RPN у результаті надає ймовірність об’єкта за кожною рамки бути класифікованим як основне зображення чи задній фон. Здійснюється регресія рамки для того, щоб покращити рамки-якорі у кожній локації.

8. Алгоритм YOLO (You Only Look Once)

Для алгоритму YOLO виявлення – це проста проблема регресії, яка вирішується взяттям початкового зображення і вивченням властивостей класів і координат рамок біля об’єктів. За алгоритмом YOLO кожне зображення поділяється на сітки розміром SxS і кожна сітка у результаті надає N рамок і відповідні ймовірності, що рамка містить об’єкт, без визначення класу. Можливе поєднання розпізнавання класів і рамок одночасно, проте більшість результуючих рамок містять непотрібну інформацію і у процесі розпізнавання не мають враховуватися.

CNN у такому випадку працює лише один раз. Таким чином, YOLO – це дуже швидкий алгоритм, який може використовуватися у часі, наближеному до реального. Відмінністю алгоритму YOLO також є те, що розглядається ціле зображення на відміну від регіонів у попередньо розглянутих методах.

Обмеженням YOLO є те, що алгоритм класифікує один тип класу у межах однієї рамки.

10. Алгоритм SSD (Single Shot Detector)

Алгоритм SSD досягає балансу між точністю і часом виконання. SSD передбачає роботу згорткової мережі на вхідному зображенні лише один раз. Після цього вихідну мапу властивостей оброблює невелика згорткова мережа розміром 3x3 для передбачення рамок і класифікації об’єктів. SSD також використовує рамки-якорі із різним співвідношенням сторін, подібно до алгоритму швидших R-CNN. Що стосується обробки масштабу, SSD видає результуючі рамки після роботи багатьох згорткових шарів. Після кожного шару, що працює на різних масштабах, можливо знаходити об’єкти різних масштабів.

11. Порівняння ефективності алгоритмів

Найбільш точний результат розпізнавання отримується алгоритмом швидших R-CNN, проте якщо для виконання задачі існують обмежені обчислювальні можливості, кращим варіантом є SSD. Якщо точність є менш пріоритетною, ніж швидкість виконання, YOLO буде найкращим варіантом. Для поточного розпізнавання SSD також є непоганим варіантом.

Порівняння точності розпізнавання і швидкості наведено на рисунку 2.

Рис. 2. Порівняння точності і швидкості виконання алгоритмів виявлення об'єктів

Висновки

У результаті для досягнення цілей статті розглянуто основні задачі розпізнавання образів, детально розглянуто задачу виявлення об'єктів, проблеми виявлення об'єктів та методи виявлення об'єктів. Розглянуто методи використання HOG, R-CNN, SPP-net, швидких R-CNN, швидших R-CNN, а також алгоритми YOLO і SSD. Розглянуто порівняння найбільш продуктивних алгоритмів у характеристиках швидкості виконання і отриманої точності розпізнавання, а саме алгоритмів швидких R-CNN, швидших R-CNN, YOLO і SSD.

Алгоритм швидших R-CNN є найкращим у показнику точності розпізнавання, YOLO демонструє найкращий результат за швидкістю розпізнавання і SSD є балансом між швидкістю виконання та точністю розпізнування. У залежності від поставленої задачі можливе використання одного з цих алгоритмів.

Список використаних джерел

1. A gentle introduction to computer vision (Стаття) / Machine Learning Mastery. – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://machinelearningmastery.com/what-is-computer-vision/
2. Machine Learning and Pattern Recognition (Стаття) / DZone. – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://dzone.com/articles/machine-learning-and-pattern-recognition
3. What is object detection? Introduction to YOLO algorithm (Стаття) / Appsilon. – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://appsilon.com/object-detection-yolo-algorithm/
4. Zero to Hero: guide to object detection using deep learning: faster R-CNN, YOLO, SSD (Стаття) / CV Tricks. – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

УДК 004.023

ЧЕРНЕНЬКИЙ А.Ю.,
ОЛІЙНИК Ю.О.

РОБОТА ІЗ ЛОКАЦІЄЮ У МОБІЛЬНИХ ЗАСТОСУНКАХ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

У цій статті розглянуто засоби взаємодії із локацією користувача у мобільних застосунках доповненої реальності. Актуальність і цілі дослідження сформульовано. Описано головні типи взаємодії із простором у застосунках доповненої реальності. Описано методи реалізації доповненої реальності для мобільних платформ. Описано популярні методи взаємодії із локацією користувача і взаємодії із простором.

КЛЮЧОВІ СЛОВА: ДОПОВНЕНА РЕАЛЬНІСТЬ, ГЕОЛОКАЦІЯ, МОБІЛЬНИЙ ЗАСТОСУНОК, МАРКЕР, GPS

This article is concerned with the ways of interaction with user geolocation in mobile augmented reality applications. The relevance and goals of the research are formulated. The main types of interaction with the environment in AR applications are described. The methods of augmented reality implementation within mobile platforms are described. The popular methods of interaction with user location and interaction with environment are described.

KEYWORDS: AUGMENTED REALITY, GEOLOCATION, MOBILE APPLICATION, MARKER, GPS

1. Вступ

Доповнена реальність (augmented reality, AR), — термін, що позначає всі проекти, спрямовані на доповнення реальності будь-якими віртуальними елементами. Доповнена реальність — складова частина змішаної реальності (англ. mixed reality), в яку також входить «доповнена віртуальність» (коли реальні об'єкти інтегруються у віртуальне середовище). Користувачами доповненої реальності є сфера розваг, медицина, освіта, військові технології, ремонтні служби, авіація, експлуатація автомобіля, служби безпеки, транспортні компанії, будівництво тощо. [1]

Застосунки доповненої реальності стають все більш поширеними. Недоліком багатьох існуючих прикладів застосування технології AR у мобільних застосунках і поточним обмеженням є неточність геолокації користувача. Без даних про знаходження користувача або із помилково визначеними даними важко розташувати об’єкти у цифровому просторі достатньо точно для виконання поставленої задачі.

Поставлена задача: дослідити існуючі методи взаємодії із геолокацією користувача у AR-застосунках.

Задача поділяється на наступні підзадачі:

4. Дослідження типів взаємодії із простором у застосунках доповненої реальності.
5. Дослідження засобів реалізації доповненої реальності на сучасним мобільних платформах
6. Дослідження методів взаємодії із простором у доповненій реальності

2. Типи застосунків доповненої реальності

Існує декілька ключових питань, які розробнику варто задати перш ніж дати користувачеві взаємодіяти із цифровим світом. Який контент має відображуватися у часі, наближеному до реального, на камері мобільного пристрою? Де саме має розташовуватися контент? Відповідь на ці питання залежить від типу застосування доповненої реальності у конкретному мобільному застосуванні. Кожен з них потребує різні типи взаємодії з користувачем.

У деяких випадках потрібно знати, на що саме дивиться користувач. Такий засіб використання доповненої реальності має назву “доповнена реальність на основі маркерів” (marker-based augmented reality).

У випадку такого застосування цифровий світ є прив’язаним до міток у реальному світі. Прикладом є розміщення освітньої анімації на зображенні у книзі. У цьому прикладі камера і користувач направлені на конкретну сторінку книги, і пристрій має спочатку розпізнати, на яку сторінку книги його направлено. Цього можна досягти розміщенням зображення, що сильно відрізняється, або фігури на сторінці. Це зображення або фігура буде розпізнано і анімація може починатися одразу, прив’язана до потрібного місця сторінки. Користувач також може переміщати книгу, але віртуальне зображення має бути “прив’язаним” до реальної поверхні сторінки. Маркером є відмінне зображення або фігура, яку можна розпізнати пристроєм. Маркером може виступати будь-який об’єкт, який має унікальні візуальні риси. Особливо ефективними є зображення з великою кількістю кутів. Зазвичай як маркери використовують логотипи, брошури.

В інших випадках потрібно лише розміщувати 3D-моделі доповненої реальності, що можливо зробити без маркера. Такий тип застосування технології має назву “доповнена реальність без маркерів” (markerless AR).

Прикладом такого застосування є розміщення віртуальних об’єктів у кімнаті. Для такого застосунку користувач має самостійно вирішувати, де розмістити віртуальний об’єкт. Також вказаний тип взаємодії із простором підходить для інтерактивних застосунків, де у віртуальному просторі розміщено персонажа і він взаємодіє із користувачем. Подібний підхід ефективний, якщо застосунок не має прив’язуватись до мітки у реальному світі. Це означає, що віртуальний об’єкт буде “висіти” у повітрі, проте можливо реалізувати розміщення 3D-моделі на площину для покращення реалізму.

У деяких випадках потрібно обробляти локальну інформацію (напрямки руху, дорожні знаки тощо). У такому випадку потрібно знати геолокацію користувача. Такий тип застосування має назву “доповнена реальність на основі локації” (location-based AR). Приклад застосунку доповненої реальності на основі локації продемонстровано на рисунку 1.

У подібних застосунках віртуальний світ є відповідним реальному. У AR-застосунках на основі локації контент доповненої реальності прив’язаний до специфічної локації. Прикладом є застосунок, у якому камера наведена на вулицю і при досягнені певного пункту демонструється віртуальний знак із назвою вулиці. Популярним прикладом застосунку, заснованому на локації у доповненій реальності, є Pokemon Go.

Розміщення віртуальних об’єктів у реальному світі є корисним для широкого спектру застосунків, від напрямків руху із віртуальними підказками і назвами вулиць до пошуку віртуальних скарбів і віртуальних туристичних турів. Стає можливим розміщувати віртуальні об’єкти на фізичних, але існує проблема розміщення AR-об’єктів саме у потрібних місцях. Для цього використовуються датчики мобільних пристроїв, такі як GPS для визначення локації, сенсори IMU (компас, акселерометр, гіроскоп) для визначення орієнтації.

Серед проблем доповненої реальності існує також задача орієнтування у приміщеннях, проте рішення цієї проблеми активно розроблюються. [2]

3. Засоби реалізації доповненої реальності на мобільних платформах

Найбільш популярними SDK доповненої реальності для мобільних платформ є Vuforia, ARKit і ARCore.

Vuforia – це платформа доповненої реальності і інструментарій розробника програмного забезпечення доповненої реальності. Vuforia використовує технології комп’ютерного зору, а також відстежування пласких зображень і простих об’ємних реальних об’єктів, наприклад, кубічних, у реальному часі. Vuforia також надає можливості розпізнавання тексту і циліндрічних маркерів. Щодо SDK для мобільних платформ, Vuforia має SDK для нативних застосунків Android і iOS, а також SDK для Unity. Vuforia використовує доповнену реальність на основі маркерів. [3]

Рис.1. Приклад доповненої реальності на основі локації

ARKit – це SDK для доповненої реальності для платформи iOS. На відміну від Vuforia, ARKit пропонує підхід доповненої реальності без маркерів. SDK містить візуальну інерційну одометрію (VIO) для точного відстеження середовища, підтримку зображень 1080p HD, розуміння сцени та оцінку освітлення, потужні пристрої з вдосконаленими датчиками камери.

ARCore – це SDK для доповненої реальності без маркерів від Google. На відміну від попереднього проекту доповненої реальності Tango, ARCore не вимагає додаткових пристроїв, щоб реалізувати доповнену реальність. Перевагою ARCore є зберігання даних локалізації реального світу у 3D. Також ARCore пропонує можливість робити "якорі" об'єктів (розміщувати цифрові об'єкти точно), виявляти горизонтальні площини і слідкувати за рухами пристрою (позиція і орієнтація). Найбільшою проблемою ARCore є те, що більша частина смартфонів із платформою Android його не підтримує.

Wikitude – це один з найбільш поширених комерційних кросплатформених SDK для розробки застосунків доповненої реальності. Фреймворк працює не тільки для мобільних пристроїв, але і для розумних окулярів, дозволяє одночасно використовувати доповнену реальність на основі маркерів і без маркерів. Також Wikitude включає розпізнавання зображень і їх відстеження, виявлення об'єктів і їх відстеження, 3D-відстеження без маркерів і геолокацію. [4]

4. Використання датчиків систем локалізування у реальному часі

Використання систем локалізування у реальному часі (real-time location system) передбачає взаємодію із датчиками інтернету речей. Це дозволяє досягнути великої точності у AR-застосунках, зокрема у застосунках для взаємодії з простором всередині приміщень.

Існує декілька видів альтернативних технологій RTLS. Серед них – інфрачервоні RTLS, WiFi RTLS, ультра широкосмугові RTLS, використання RFID тощо. Узагальнено кожна з технологій працює з фізичними датчиками, які розміщено у приміщеннях або на територіях, де має здійснюватися визначення локації. [5]

5. Використання рішень одночасного локалізування і картографування

Для задач взаємодії із локацією користувача можливе використання мап навколишнього середовища. Якщо місце проведення взаємодії із локацією може змінюватися, варто розглянути складання мапи протягом взаємодії користувача із застосунком доповненої реальності.

Одночасне локалізування і створення мапи (simultaneous localization and mapping, скорочено SLAM) – це алгоритмічна обчислювальна задача побудови і оновлення мапи невідомого оточення з одночасним відстежуванням місцеположення рухаючись по ньому.

Задачу SLAM можна сформулювати наступним чином: для послідовності даних спостереження сенсору o_t за дискретні проміжки часу t розрахувати і визначити розташування агента x_t і мапу оточення m_t.

Функція переходу оновлення розташування агента може бути сформульована таким чином:

$$P(x_t|o_{1:t}, m_t) = \sum_{m_{t-1}} P(o_t|x_t, m_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}|m_t, o_{1:t-1})/Z$$

Оновлення мапи відбувається поступово наступним шляхом:

$$P(m_t|o_{1:t}, m_t) = \sum_{x_t} \sum_{m_t} P(m_t|x_t, m_{t-1}, o_t) P(m_{t-1}, x_t|o_{1:t-1}, m_{t-1})$$

Як для більшості задач наближення, рішення можна знайти при наближенні двох змінних до локального оптимального рішення.

Існує декілька алгоритмів, що розв'язують її у кінцевий час для певних умов. До популярних рішень відносять фільтр часток, розширений фільтр Калмана і GraphSLAM.

Існують open-source рішення, націлені саме на використання у доповненій реальності. Ці методи також реалізовано в безпілотних автомобілях і літаючих засобах, автономних підводних апаратах тощо.

6. Використання існуючої 3D-моделі оточення

Для кращого орієнтування користувача чи цифрових об'єктів у населених пунктах при використанні застосунку доповненої реальності використовуються 3D-моделі світу. Таким чином, проблема орієнтування об'єктів і користувача у реальному світі трансформується у проблему орієнтування всередині 3D-моделі.

Існує ряд комерційних рішень, які надають 3D-моделі світу і окремих міст. Серед них популярними є WRLD, AR GPS Compass Map, MapBox. Рішення є кросплатформеними.

Висновки

У результаті для досягнення цілей статті розглянуто засоби взаємодії із локацією користувача та взаємодії із простором у мобільному застосуванні із використанням доповненої реальності.

Засоби RTLS, SLAM-рішення і використання 3D-моделі простору значно покращують орієнтування у просторі і надають значно більше можливостей, ніж просто використання GPS. Проте всі вказані засоби, окрім відкритих SLAM-рішень, потребують закупівлі додаткового інструментарію. Існують SLAM-рішення із відкритим кодом, які розвиваються спільнотами програмістів, проте SLAM-рішення не підходить для ряду застосувань доповненої реальності на мобільних платформах.

Список використаних джерел

1. What is augmented reality and how does it work (Стаття) / ThinkMobiles. – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://thinkmobiles.com/blog/what-is-augmented-reality/
2. 3 different types of AR explained: marker-based, markerless and location (Стаття) / BlippAR. – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://www.blippar.com/blog/2018/08/14/marker-based-markerless-or-location-based-ar-different-types-of-ar
3. Vuforia (Стаття) / Vuforia. – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://www.vuforia.com
4. Everything you need to know to build location-based AR app (Стаття) / Vakoms. – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://blog.vakoms.com/everything-you-need-to-knowto-build-location-based-ar-app/
5. Real-time location systems overview (Стаття) / AirFinder. – Електрон. дан. (1 файл). – 2019. – Режим доступу: https://www.airfinder.com/blog/rtls-technologies/real-time-location-systems-overview-technologies

УДК 004.023

МИКОЛАЄНКО М.О. ,
КЛИМЕНКО О.М.

РОЗШИРЕНЕ ВЕБ-ЗАСТОСУВАННЯ ПОШУКУ РОБОТИ

Робота присвячена розробці розширеного веб-застосування пошуку роботи, яке буде особливо корисне для студенів та претендентів без досвіду. Веб-застосування дозволить моніторити одразу декілька сайтів для пошуку роботи, складати резюме під кожну вакансію, яка зацікавила, а також формувати план навчання для здобуття навичок, яких не вистачає.

КЛЮЧОВІ СЛОВА: робота, навчання, досвід роботи, резюме, вакансії.

The work is devoted to the development of an advanced Web application for job search which will be especially useful for students and applicants without experience. Web application will allow you to monitor multiple sites for job search, summarize each vacancy you are interested in, also it will create a training plan for skills that are lacking.

KEYWORDS: work, study, work experience, resume, vacancies.

1. Вступ

В наш час студенту інколи важко знайти роботу після закінчення університету, оскільки роботодавцям потрібні працівники вже з досвідом роботи. Вони не хочуть витрачати гроші і час на недосвідчених працівників, звісно, якщо це не компанії, які шукають собі людей на стажування. Для того, щоб студенту знайти роботу йому необхідно скласти резюме і відправити на всі бажані вакансії. Звісно в наш час є багато веб-сервісів, які дозволяють знайти роботу, а також скласти резюме, але немає єдиного сервісу, який дозволить шукати вакансії одразу на декількох цих сайтах, а також, який допоможе не просто створювати резюме, а на основі побажань користувача будувати спеціальне резюме під кожну вакансію окремо, а також скласти для студента план навчання, якщо не вистачає певних навичок для роботи.

2. Огляд можливих технологій

Для реалізації веб-застосування використовуємо мову програмування PHP, оскільки вона добре поєднується з HTML тегами, та базу даних MySql.

PHP (англ. PHP: Hypertext Preprocessor) є однією з найпоширеніших мов, що використовуються у сфері веб-розробоки. Ця мова може бути вбудованою безпосередньо в html-код сторінок, які, в свою чергу коректно будуть оброблені PHP -інтерпретатором.

MySQL - це система управління реляційними базами даних. У реляційній базі даних дані зберігаються в окремих таблицях, завдяки чому досягається виграш у швидкості й гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість поєднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мова структурованих запитів, що використовується для доступу до баз даних.

3. Робота програми

Програма повинна працювати з API деяких сайтів, які спеціалізуються на пошуку роботи, щоб брати інформацію про вакансії. Потім інформацію потрібно записувати в базу данних, щоб кожного разу при пошуку роботи користувачем не навантажувати сервер, а видавати йому актуальні вакансії за останній час.

Потрібно розробити скрипт який буде автоматично брати інформацію з API та записувати в базу данних тоді, коли навантаження на сервер може бути мінімальним (наприклад, вночі) та перевіряти, чи немає цієї ж вакансії вже у базі данних.

4. Переваги веб-додатку

Щоб наш додаток мав ще більше переваг перед популярними сервісами, необхідно реалізувати не просто конструктор резюме, а

такий, який буде обробляти інформацію про навички користувача, а також створювати резюме спеціально під бажані вакансії. А щоб остаточно переконати шукачів роботи обрати саме наш веб-додаток для пошуку майбутнього місця роботи, потрібно реалізувати можливість побудови розкладу вивчення нових навичків, або рекомендації, де можна отримати потрібні знання, в тому випадку, якщо користувач не володіє достатніми знаннями, але бажає отримати роботу по даній вакансії.

5. Додатковий функціонал

Також у користувача повинна бути можливість зберегти резюме, щоб в майбутньому мати змогу його змінити, або завантажити, а для цього необхідно зробити реєстрацію користувачів і особистий кабінет, де будуть зберігатися створені резюме, та можливо переглядати історію пошуків, або історію вдосконалення навичок користувача. Можна робити аналітику щодо популярності вакансій/спеціальностей за неділю, місяць і т,д,

Висновок

Планується що такий веб-додаток зможе не тільки допомогти студентам у пошуку роботи своєї мрії, але й отримати інформацію (статистику) щодо популярності тої чи іншої спеціальності у даний час, що може допомогти викладачам зробити правильний вибір у викладі матеріалу, якщо викладач не знає що буде більш корисним для вивчення студентами у данний час.

Література

1. Що таке PHP? [Електронний ресурс] – Режим доступу до ресурсу: http://phpist.com.ua/php/4-whatphp.
2. PHP + MySQL [Електронний ресурс] – Режим доступу до ресурсу: http://sites.znu.edu.ua/webprog/lect/1222.ukr.html
3. MySQL :: MySQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/.
4. PHP: Что такое PHP? - Manual [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/manual/ru/intro-whatis.php.
5. PHP: Руководство по PHP - Manual [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/manual/ru/.
6. Коротко про API і його тестування — дізнайтеся на базовому рівні [Електронний ресурс] – Режим доступу до ресурсу: https://www.quality-assurance-group.com/korotko-pro-api-jogo-testuvannya/

УДК 004.02

БОГДАНЕНКО М.О.

ВИКОРИСТАННЯ ЛІНІЙНОЇ РЕГРЕСІЇ ДЛЯ ВИБОРУ СТРАТЕГІЇ МАЙНІНГУ КРИПТОВАЛЮТИ У СИСТЕМАХ АВТОМАТИЗАЦІЇ ПРОЦЕСУ МАЙНІНГУ

У цій роботі буде розглянуто використання лінійної регресії, результати створення моделі якої можуть бути використані у подальшому аналізу і дати відповідь на питання «яку криптовалюту має сенс майнити у даний момент часу?». Оскільки створення моделі вирішується математично, то побудову лінійної регресії та подальший аналіз можна використовувати у системах автоматизації процесу майнінгу. Задача побудови лінійної регресії розглядається на малих інтервалах з вибірки даних, оскільки тоді створений з меодлі регресії прогноз враховує поточну динаміку ринку.

КЛЮЧОВІ СЛОВА: МАЙНІНГ, ЛІНІЙНА, РЕГРЕСІЯ, СТРАТЕГІЯ, КРИПТОВАЛЮТА, ТРЕНД, АВТОМАТИЗАЦІЯ

In this paper, we will consider the use of linear regression, the results of which the model can be used for further analysis and give an answer to the question "what kind of cryptography it makes sense to own at the given time?". Since the creation of the model is solved mathematically, then the construction of linear regression and further analysis can be used in the systems of automation of the process of landing. The task of constructing linear regression is considered at small intervals from the data sample, since the forecast generated from the regression modal then takes into account the current market dynamics.

KEYWORDS: MINING, LINEAR, REGRESSION, STRATEGY, CRYPTOCURRENCY, TREND, AUTOMATION

1. Вступ

Розвиток кібернетичних технологій призвів до багатьох змін у суспільстві. Порівнюючи стан суспільства із його більш ранніми станами: 20, 30, 50 років тому, можна відмітити радикальні зміни у економіці, виробництві та науці. Ці зміни також зачепили структуру суспільства, характер взаємовідносини між людьми, вплинули як на їх роботу, так і на повсякденне життя.

Сьогодні вже не можливо знайти людину, котра проживає у розвинутій країні, і котра не знала б про соціальні мережі, чи інтернет, котра не використовувала б їх на роботі. Але з появою нових технологій з'являються і нові проблеми, так виникла серйозна проблема кібернетичного захисту. Власне, ця проблема існувала завжди, проте у більш ранніх формах це була проблема криптографії, тобто, шифрування.

З появою і розвитком електронних мереж виникла також можливість передавати гроші у електронному вигляді, що дуже зручно і швидко, але без захисту - ризиковано. Захист операцій передачі грошей – ще одна із галузей кібернетичного захисту. Спочатку такі операції проводились банківськими структурами, що давало захищеність операцій, проте через централізованість процесу ці операції ніколи не були приватними: принаймні одна «зайва» сторона, тобто, банк, знала про їх існування.

Для вирішення цієї проблеми були винайдені захищені електронні гроші - криптовалюти. Через децентралізацію, тобто розподіленість мережі криптовалют на багатьох «нодах» - обчислювальних машинах, та через відсутність будь-яких даних про користувачів, окрім номерів гаманців, проблема приватності була вирішена, а проблема захищеності операцій вирішується самою структурою системи криптовалют. Та сама децентралізація не дозволяє одному із користувачів еміссувати власні одиниці криптовалюти і використовувати їх. Мережа криптовалюти працює не з фактом «наявності», скільки із фактом «доступності» грошей, тобто, якщо рахунок має якусь кількість грошей, які на нього надійшли, і якусь кількість грошей, які із нього було знято, то різниця цих двох чисел і є «доступні» гроші. Якщо гроші доступні – операції із ними можливі. Якщо ні – то операції буде скасовано мережею.

Для підтримки роботи мережі необхідне існування «нод» - це обчислювальні машини, які забезпечують децентралізацію та безпеку мережі за рахунок перевірки транзакцій(операцій).

Як було сказано, «ноди» - це обчислювальні машини, які забезпечують корректне функціонування мережі криптовалюти. Їх зазвичай називають «майнерами» або «майнінг-фермами», а сам процес обчислень зветься «майнінгом». Це пов'язано у першу чергу з тим, що майнери здійснюють обчислення, за рахунок чого відбувається емісія грошей і підтвердження транзакцій. Випущені гроші нараховуються на рахунки власників майнерів, які здійснили правильні обчислення. Назва процесу походить від англійського слова «mining» – видобуток, за цією логікою і було названо процес, оскільки він буквально видобуває гроші.

Варто згадати про важливу особливість мереж криптовалют. Чим більш потужна мережа(чим більше обчислювальних здатностей задіяно у обчисленнях у мережі), тим більша складність. Це потрібне для того, щоб не відбувалося гіперемісії, тобто щоб випуск нових одиниць криптовають був рівномірний. Але якщо майнерів мало - гіпоеміссії не відбувається, оскільки мережа підлаштовує складність під задіяні обчислювальні ресурси. Таким чином

підтримується ціна криптовалют і безпека операцій.

Враховуючи це, не всі майнери, особливо із невеликими обчислювальними здатностями можуть принести власнику винагороду за майнінг. Це призводить до зниження зацікавленості людей у майнінгу, через що ноди перестають функціонувати, знижується безпека операцій, з'являється ризик підробки криптовалют.

Враховуючи це, для підвищення інтересу до майнінгу має сенс збільшити шанс на отримання винагороди від майнінгу для кожного власника майнінг-ферм. Оскільки не має сенсу докупати нове обладнання – через зміну складності мережі, має сенс об'єднати зусилля майнерів, і відповідно до вкладу розподіляти між ними винагороду. Таким чином буде підтримуватись інтерес, відповідно, можлива поява нових зацікавлених у майнінгу осіб. У подальшому це призведе до підвищення безпеки криптовалют, до збільшення використання криптовалют і, відповідно, до збільшення їх ціни.

Як було сказано, має сенс об'єднувати зусилля власників майнінг-ферм для збільшення їх прибутку, отже, підтримки або збільшення їх зацікавленості. Проте, не всі криптовалюти однаково коштують, і інколи майнінг веде лище до збитків. Тоді має сенс майнити іншу криптовалюту, для чого власники майнінг-ферм мають їх переналаштовувати в залежності від стану ринку. Хоч таким чином вони отримують прибуток, майнінг – чесний процес, адже він забезпечує безпеку операцій для всіх користувачів криптовалюти, навать тих, хто не майнить, а лише користується електронними грошами.

Для об'єднання зусиль можна скористатись системами управління процесом майнінгу, які будуть автоматично обирати більш вигідний варіант для майнінгу. Хоч при цьому і зменшується безпека деяких криптовалют, система криптовалют в цілому виграє, оскільки не зменшується кількість власників майнерів і в ній зберігається баланс: з занадто захищених криптовалют майнери переключатися на менш захищені.

У цій роботі буде розглянуто частина методу вибору стратегії майнінгу, що використовується у системах автоматизації процесу майнінгу. Конкретно, тут розглядається метод лінійної регресії, який використовується для аналізу цін криптовалют і для подальшого прийняття рішень щодо стратегій.

2. Лінійна регресія

Це метод моделювання залежності між скаляром y та векторною змінною X. У випадку, якщо змінна X також є скаляром, а не вектором, регресію називають простою.

При використанні лінійної регресії взаємозв'язок між даними моделюється за допомогою лінійних функцій, а невідомі параметри моделі оцінюються за вхідними даними. Подібно до інших методів регресійного аналізу лінійна регресія повертає розподіл умовної імовірності у в залежності від X, а не розподіл спільної імовірності у та X, що стосується області мультиваріативного аналізу.

При розрахунках параметрів моделі лінійної регресії як правило застосовується метод найменших квадратів (МНК), але також можуть бути використані інші методи. Так само метод найменших квадратів може бути використаний і для нелінійних моделей. Тому МНК та лінійна регресія хоч і є тісно пов'язаними, але не є синонімами.

Зазвичай регресійна модель визначається у вигляді

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + u$$

де y – шукана змінна, x_1,x_2,...x_k – незалежні змінні, u – похибка, котру можна пропустити [1].

У нашому випадку модель залежить лише від однієї змінної x, тому ця модель є простою, і приймає вигляд:

$$y = \beta_0 + \beta_1 x$$

Як було сказано, для знаходження коефіцієнтів біля змінних може використовуватиметься один з багатьох методів, в даному випадку розглядатиметься метод найменших квадратів.

Метод найменших квадратів – це метод знаходження розв'язку надлишково-визначеної системи. Він часто

застосовується в регресійному аналізі. На практиці найчастіше використовується лінійний метод найменших квадратів, що використовується у випадку системи лінійних рівнянь. Наш випадок теж можна привести у вигляд системи лінійних рівнянь. Зокрема важливим застосуванням у цьому випадку є оцінка параметрів у лінійній регресії, що широко застосовується у математичній статистиці і економетриці [2].

Для знаходження моделі нам потрібно знайти значення коефіцієнтів. Для цього нам потрібно обчислити наступні формули, використовуючи наші значення з вибірки даних:

$$\beta_0 = \bar{Y} - \beta_1 \bar{X},$$

$$\beta_1 = \frac{n\sum_i x_i y_i - \sum_i x_i \sum_i y_i}{n\sum_i (x_i - \bar{X})},$$

де $\bar{Y}$ - середнє значення вибірки значень ціни, $\bar{X}$ – середнє значення пронумерованих моментів часу, x_i – номер момента часу з вибірки, y_i – значення ціни у момент часу x_i з вибірки, n – розмір вибірки.

3. Постановка задачі вибору криптовалюти у загальному вигляді

Нехай маємо множину криптовалют, кожна з яких представлена у вигляді часового ряду $Y_i = \{y_{i1}, y_{i2}, \dots, y_{im}\}$, де i – унікальний номер криптовалюти у списку, m – довжин ряду. Цей ряд містить значення y_{ij}, що являються скалярними і відповідають певним значенням x_i, тобто y залежна від x.

Потрібно на основі поточної ціни і тренду криптовалюти визначити, яку криптовалюти зі списку вигідніше майнити.

Проаналізувавши ці дані необхідно поставити у відповідність кожній криптовалюті певну оцінку прибутковості і визначити, у якої криптовалюти ця оцінка максимальна. Криптовалюта з максимальною оцінкою і є шуканою криптовалютою для майнінгу.

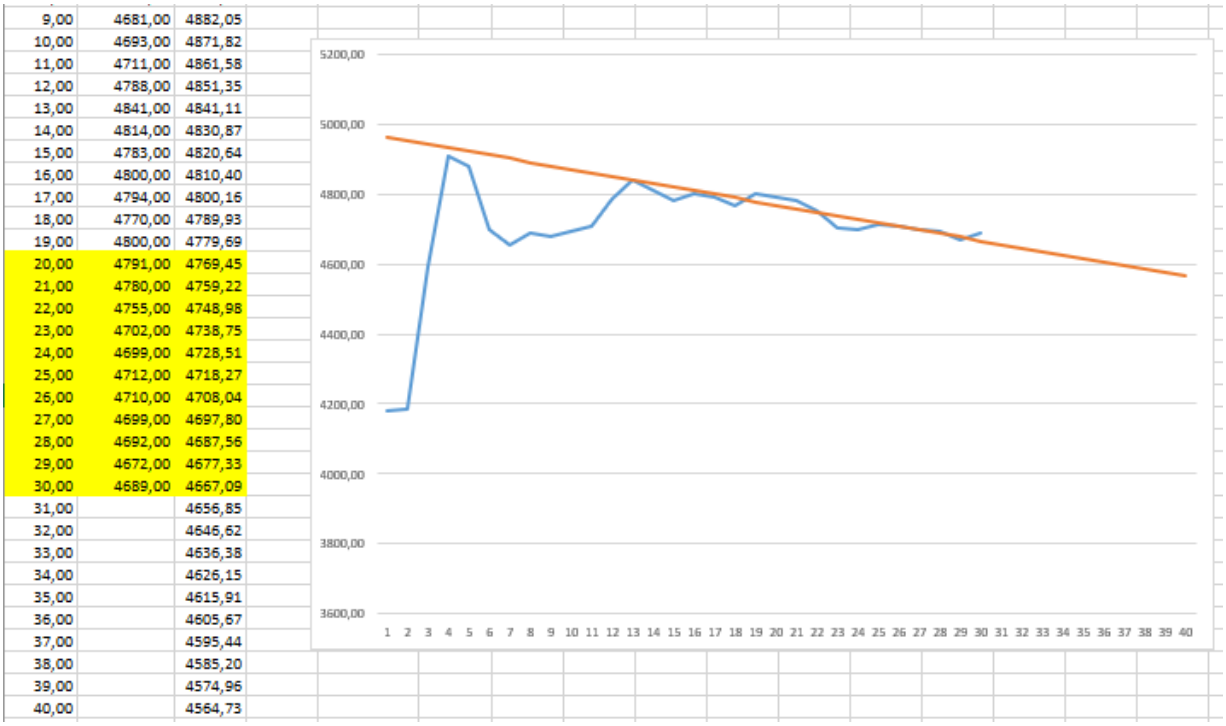
4. Ціль використання лінійної регресії

Мета використання лінійної регресії полягає у тому, щоб визначити для кожної криптовалюти значення коефіцієнта β_1, за допомогою якого буде спрогнозовано тренд. Наприклад, нехай ми отримали з начення коефіцієнтів методом найменших квадратів, якщо накласти отриману модель на графік значень часового ряду, можна отримати графічне відображення тренду на графіку часового ряду. Логічно використовувати у розрахунках не всі отриманні значення ряду, а лише якусь частину з останніх, оскільки ринок змінюється і вплив давніших факторів повністю або частково нівелюється новішими. Так, наприклад, можна спостерігати, що значення ціни певної криптовалюти, якщо аналізувати лише деяку останню частину ряду, спадає, але якщо брати весь отриманий ряд, то значення коефіцієнта буде додатнім, з чого можна отримати невірний висновок, що ціна зростає (рис.1). Жовтим кольором позначено дані, які було використано для аналізу.

Ефективність методу можна довести проілюструвавши на прикладі, що наведено на рисунку 2.

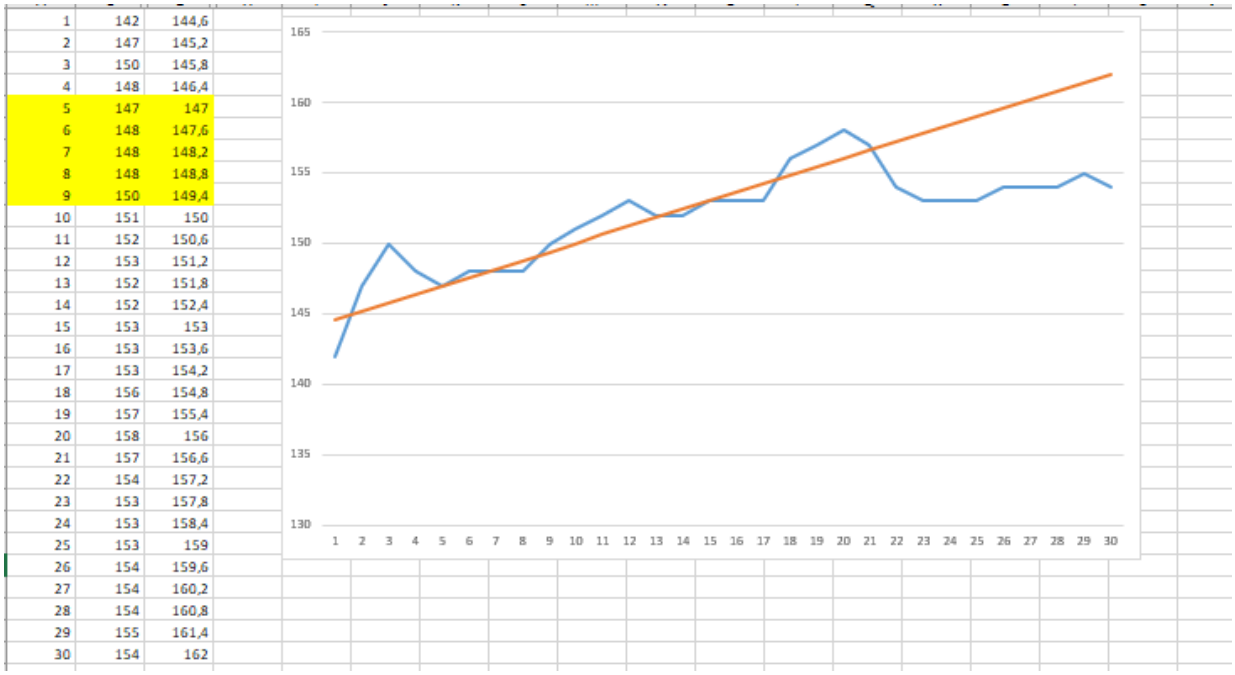
У ході аналізу, для наглядності, було обрано випадковий діапазон даних і створено для нього модель лінійної регресії. Як видно, отримана лінійна модель зростаюча, і подальші значення, які можна спостерігати на графіку також більші за попередні.

Ринок криптовалют непередбачуваний і швидко змінюється, проте малий проміжок аналізу дозволяє побудувати тренд до обраної криптовалюти для поточної ситуації на ринку, що дозволить отримати достовірний результат. Варто вказати, що при від'ємному значенні β_1 тренд є спадаючим, тобто ціна криптовалюти також спадає і, скоріше за все, спадатиме у майбутньому. По $\beta_1 > 0$, тобто по зростаючому тренду ми можемо зробити висновок, що ціна буде надалі зростати.



9,00	4681,00	4882,05
10,00	4693,00	4871,82
11,00	4711,00	4861,58
12,00	4788,00	4851,35
13,00	4841,00	4841,11
14,00	4814,00	4830,87
15,00	4783,00	4820,64
16,00	4800,00	4810,40
17,00	4794,00	4800,16
18,00	4770,00	4789,93
19,00	4800,00	4779,69
20,00	4791,00	4769,45
21,00	4780,00	4759,22
22,00	4755,00	4748,98
23,00	4702,00	4738,75
24,00	4699,00	4728,51
25,00	4712,00	4718,27
26,00	4710,00	4708,04
27,00	4699,00	4697,80
28,00	4692,00	4687,56
29,00	4672,00	4677,33
30,00	4689,00	4667,09
31,00		4656,85
32,00		4646,62
33,00		4636,38
34,00		4626,15
35,00		4615,91
36,00		4605,67
37,00		4595,44
38,00		4585,20
39,00		4574,96
40,00		4564,73

Рис. 1. Приклад вхідних даних та результатів аналізу. Для аналізу взято кінець вибірки.



1	142	144,6
2	147	145,2
3	150	145,8
4	148	146,4
5	147	147
6	148	147,6
7	148	148,2
8	148	148,8
9	150	149,4
10	151	150
11	152	150,6
12	153	151,2
13	152	151,8
14	152	152,4
15	153	153
16	153	153,6
17	153	154,2
18	156	154,8
19	157	155,4
20	158	156
21	157	156,6
22	154	157,2
23	153	157,8
24	153	158,4
25	153	159
26	154	159,6
27	154	160,2
28	154	160,8
29	155	161,4
30	154	162

Рис. 2. Приклад вхідних даних та результатів аналізу. Для наглядності аналізу взято дані із середини вибірки.

5. Метод вибору стратегії майнінгу

Стратегія майнінгу – це рішення щодо того, яку криптовалюту із розглянутого списку доцільно обрати для подальшого майнінгу. Для цього ми використовуємо отриману на попередніх кроках модель. Сама по собі модель дає достатні дані для прийняття рішення, проте для покращення результатів можна допрацювати метод. Так, наприклад, можна використовувати лише значення коефіцієнта β_1 і обирати ту криптовалюту, значення у якої максимальне, але цей метод можна покращити, врахувавши значення ціни

криптовалюти. Тоді вибір паде на криптовалюти, що мають більшу ціну і у яких відповідний $\beta_1 > 0$ (зростаючий тренд).

Також існує група методів, що будують прогноз прибутку по кожній криптовалюті і обирають ту, що має більше значення прогнозованого прибутку. У широкому сенсі вони мало відрізняються від попередніх, оскільки мають той самий базис.

Також, базуючись на прогнозованому прибутку і даних від автоматизованої системи управління майнінгом можна обирати криптовалюту з врахуванням обчислювальних здатностей і потужності мережі. Тобто, якщо криптовалюта має високу ціну, але шанс на отримання її дуже низький, то не має сенсу майнити цю криптовалюту, адже краще обрати ту, що має меншу ціну і менше зростає, проте шанс на отримання цієї криптовалюти значно більший.

Висновок

У роботі розглянуто і описано побудову лінійної регресії для використання отриманих від моделі даних у прийнятті рішень щодо стратегій майнінгу. Застосований підхід до прогнозування трендів на малих інтервалах дозволяє враховувати високу динамічність ринку криптовалют і тому дає достатньо достовірні дані. Навіть у разі хибності прогнозу, через те що розглядаються малі проміжки часу і вибір нової стратегії відбувається часто, хибний прогноз і, відповідно, рішення, не призведуть до великих втрат, пов'язаних із зменшенням ціни обраної криптовалюти. Використання лінійної регресії для вирішення цього класу задач можна вважати прийнятним, що було доведено на прикладі у розділі 4 цієї роботи.

Список літератури

1. Лінійна регресія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D0%BD%D1%96%D0%B9%D0%BD%D0%B0_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%96%D1%8F
2. Метод найменших квадратів [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BD%D0%B0%D0%B9%D0%BC%D0%B5%D0%BD%D1%88%D0%B8%D1%85_%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D1%96%D0%B2

УДК 004.021

ТРУХАН Г.О.
ПРОСКУРА С.Л.

АЛГОРИТМИ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ІГРОВОГО КОНТЕНТУ НА ПРИКЛАДІ ГЕНЕРАЦІЇ МІСТА НА 3D-СЦЕНІ ЗАСОБАМИ UNREAL ENGINE 4

В даній статті розглянута процедурна генерація міста на 3D сцені засобами Unreal Engine 4. Демонструється використання математичних алгоритмів для створення реалістичної процедурної генерації міста, міських парків, декоративних елементів. Використання даної системи дозволить розробникам відео-ігр на Unreal Engine 4 значно скоротити час створення локацій.

КЛЮЧОВІ СЛОВА: UNREAL ENGINE 4, ПРОЦЕДУРНА ГЕНЕРАЦІЯ, ІГРОВИЙ РІВЕНЬ, КАРТА, 3D-СЦЕНА, АЛГОРИТМ КОМІВОЯЖЕРА, ДІЛЕННЯ ПРЯМОКУТНИКІВ, ПОБУДОВА ПАВУТИНИ.

Procedural city generation in Unreal Engine 4 was described in this article. Demonstrated the usage of mathematical algorithms in creating realistic procedural city, which includes parks and roads generation, generation and location of modular buildings, city decorative elements. Current system will allow UE4 developers reduce the time of levels (maps) creation.

KEYWORDS: UNREAL ENGINE 4, PROCEDURAL GENERATION, GAME LEVEL, MAP, 3D-SCENE, TRAVELING SALESMAN ALGORITHM, SQUARES DIVIDING, NET CREATION

1. Вступ

На сьогоднішній день індустрія розробки відеоігор активно розвивається у всіх країнах світу. Завдяки створеним умовам для розробників, останні можуть створювати якісні продукти з невеликими командами та бюджетами, або взагалі без бюджету. Сучасний ігровий редактор дозволяє навіть одному програмісту, маючи на руках всі необхідні асети для гри (моделі, звуки, музику) створити готовий для продажі продукт за відносно короткий термін.

Не дивлячись на розвиток сучасних ігрових редакторів, створення великих рівнів площею в десятки квадратних кілометрів може займати в ігрових студій найвищого рівня декілька місяців, а то і років. Вирішити цю проблему можна розробивши програмне забезпечення (або краще – розширення до професійного ігрового редактора), яке допоможе розробникам генерувати окремі частини ігрових рівнів без витрачання місяців на ручне заповнення сцен контентом.

На даний момент існує певна кількість аналогів, які вирішують подібні задачі. Процедурний генератор будинків від Ammobox Studios [1], генератор доріг від Andry K [2] і подібні аналоги спрощують розміщення на рандомізацію доріг та будинків. Проте немає відомого аналогу, який би дозволяв розробникам на Unreal Engine 4[3] генерувати комплексне місто з різноманітними елементами прикладаючи мінімум зусиль.

2. Постановка задачі

Процес створення ігрових рівнів (карт) для сучасних відеоігр вимагає багато часу, використання роботи спеціалістів у галузі 3D-арт, левел-дизайнерів та гейм-дизайнерів. Спростити цей процес можна, якщо до ігрового редактора, з яким працює команда розробників, підключити певний плагін, за допомогою якого можна згенерувати місцевість за певними алгоритмами, а потім вже змінювати отриману карту відповідно до вимог по проекту.

Користувачем системи може бути будь-який спеціаліст, що займається розробкою відеоігор на Unreal Engine 4. Найбільш вірогідні користувачі це:

- Level-дизайнер;
- Game-дизайнер;
- Environment-дизайнер;
- Програміст-розробник геймплею.

При цьому згенеровану місцевість можна використати як осново для створення фінальної (game-end) карти, так і для тестування певних ігрових механік, які важко перевірити на пустій сцені з примітивними об’єктами.

Метою даної статті є описання математичних алгоритмів, які можуть бути використані в процедурній генерації ігрового контенту на прикладі генерації міста засобами Unreal Engine 4.

У зв’язку з великою кількістю ручної роботи по створенню ігрової карти, яка, як правило, зводиться до розміщення ігрових об’єктів, актуальним є створення певного розширення для ігрового редактора, яке б могло самостійно виконати цю роботу так, щоб розробникам залишалось лише змінити певні деталі, а не проводити розробку з нуля. При цьому така система повинна надавати реалістичний результат і приймати на вхід 3D-моделі, створені розробниками для розміщення на сцені.

В контексті процедурної генерації міста, маємо декілька ключових задач, які вимагають рішення. Сюди входять:

- Генерація карти доріг декількома методами та розміщення елементів доріг (3D-моделей);
- Генерація та розміщення будинків, що складаються з модульних елементів;
- Генерація парків та їх декоративних елементів.

3. Генерація міських доріг методом розбиття прямокутників

Призначенням цієї задачі є побудова системи доріг в місті за американським типом. На етапі роботи алгоритмів карта доріг представлена матрицею. Дано:

- Розмір прямокутника, в якому будуть згенеровані міські дороги (MxN). Одиниця даної розмірності в ігровому редакторі відповідає одиниці відстані (1.0) на сцені.

- Частота розміщення елементів дороги (Freq). Даний параметр відповідає ширині та довжині елемента дороги (в даному алгоритмі елемент дороги (3D модель) має квадратну форму по осям X та Y, довільну по Z).

- Мінімальна довжина вулиці L

- Коефіцієнт розбиття (k)

Ціль задачі – оперуючи даними параметрами згенерувати матрицю m цілих чисел розмірності (M/freq)x(N/freq), елементи якої відповідають розміщенню доріг в місті. Елемент матриці m може приймати одне з наступних значень:

1) 0 – дороги не має;
2) 1 - дорога є, йде під кутом 0 град;
3) 2 – дорога є, йде під кутом 90 град;
4) 3 – перехрестя.

Нехай користувач задав, що ширина та довжина міста дорівнює M та N відповідно (в одиницях довжини реального світу), при цьому Freq – ширина та довжина квадратної клітинки, яку ми обираємо за мінімальний математичний розмір одиниці площі міста. Тоді матриця S що є математичною основою майбутньої карти доріг буде мати розмірнсть m x n, де:

$$m = \frac{M}{Freq}$$
$$n = \frac{N}{Freq}$$
$$s_{i,j} \in Z$$
$$i \in 0 \dots m-1$$
$$j \in 0 \dots n-1$$

При цьому ми маємо параметр L – мінімальний розмір вулиці, та коефіцієнт k, який буде використаний при діленні прямокутників.

Для більш детального пояснення наведений псевдокод алгоритму.

Крок 0. Згенерувати матрицю S розмірності m×n та заповнити її нулями.

Крок 1. Обрати за поточний прямокутник всю матрицю. Поточний прямокутник заданий індексами його лівого верхнього та правого нижнього краю (i_1,j_1та i_2,j_2). Почати рекурсію.

Крок 2. ЯКШО ширина АБО довжина поточного прямокутника менша за L СТОП. ІНАКШЕ:

Розділити поточний прямокутник на чотири прямокутники розрізанням поточного прямокутника на 4 частини. Індекси матриці, через які пройде вертикальний та горизонтальний розріз визначаються таким чином:

Індекс i відповідає рядку по якому буде розрізаний прямокутник.

$$i = \frac{i_1 + i_2}{2} - Rand(-L * k, L * k)$$

Індекс j відповідає стовпцю, по якому буде розрізаний прямокутник.

$$i = \frac{j_1 + j_2}{2} - Rand(-L * k, L * k)$$

Крок 2.1. По черзі передати координати початку та кінця кожного з чотирьох прямокутників до кроку 2.

Генерація міських доріг методом павутини

Для початку треба згенерувати певну кількість концентричних кіл N. Маємо радіус першого кола R_1 та коефіцієнт зміни радіусу r. Таким чином радіус кожного кола можна вирахувати за формулою:

$$R_i = R_1 * r^{i-1}$$

де i – порядковий номер поточного кола.

Тобто, маючи за коефіцієнт r число більше за одиницю, радіус кожного наступного кола буде плавно зростати.

Нехай користувач задав кількість точок на першому колі N_i та коефіцієнт зміни кількості точок n.

Тоді кількість точок на кожному колі можна розрахувати на заступною формулою:

$$N_i = N_1 * n^{j-1}$$

де j – порядковий номер поточного кола

Знову таки, маючи за коефіцієнт n число більше за одиницю, ми будемо отримувати кола з більшою кількістю точок. Самі точки на кожному колі розміщуються на однаковій відстані одне від одного, рівномірно по периметру кола.

Тепер, щоб майбутня карта доріг міста не виглядала занадто «стерильною», використовуємо параметр DotOffset та коефіцієнт k для зсуву. Таким чином, випадковий зсув для кожної точки можна розрахувати за формулою:

$$Offset = Rand\ (-DotOffset * k^{i-1}, DotOffset * k^{i-1})$$

- де Random – повертає випадкове значення в заданому діапазоні;
- i – порядковий номер поточного кола.

Останній етап роботи алгоритму – з'єднати лініями кола між собою. Для цього, для кожної точки на колі і, треба знайти таку точку на колі з номером і+1, що відстань до неї буде мінімальною.

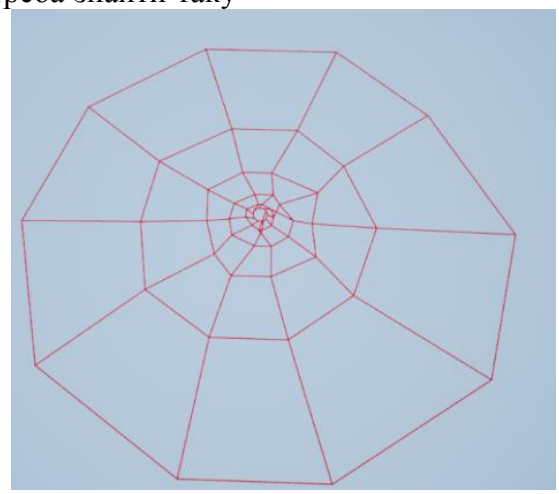


Рис. 1. - Схема згенерованої карти доріг методом павутини

4. Розміщення будинків в місті

Задачу розміщення будинків в кварталі можна вирішити наступним чином. Нехай маємо прямокутник на сцені (квартал). Тоді:

1. Згенерувати будинок обраної ширини, довжини та висоти (параметри W, L, H що згенеровані відповідно до відстані до центру міста), обрати його за поточний;

2. Розмістити будинок в центрі кварталу, обрати його (будинок) за поточний. ЯКЩО геометрія будинку пересікається з геометрією дороги - видалити будинок зі сцени, СТОП. ІНАКШЕ перейти до кроку 3;

3. Додати поточний будинок до масиву HousesArray, перемішати масив;

4. Згенерувати будинок обраної ширини, довжини та висоти, обрати його за поточний;

4.1. Обрати перший індекс HousesArray за поточний;

4.2. Спробувати розмістити поточний будинок спереду, зліва, позаду та праворуч (порядок сторін - кожен раз випадковий) будинку, що відповідає поточному індексу HousesArray (при розміщенні врахувати Offset що задав користувач);

4.3. ЯКЩО вдалося розмістити без пересікань геометрії - перейти до кроку 3. ІНАКШЕ:

4.3.1. видалити будинок що відповідає поточному індексу із масиву (але не зі сцени);

4.3.2. ЯКЩО залишились будинки в HousesArray, ТОДІ обрати наступний індекс масиву HousesArray за поточний індекс, перейти до кроку 4.2. ІНАКШЕ знищити будинок, СТОП.

Таким чином будинки починають генеруватися від центру кварталу до його країв, заповнюючи майже весь вільний простір. При цьому залишаються реалістичні "пробіли" між будинками, які в майбутньому можуть бути використані для генерації внутрішніх двориків, а "пробіли" від дороги до будинків - для кіосків, парковок.

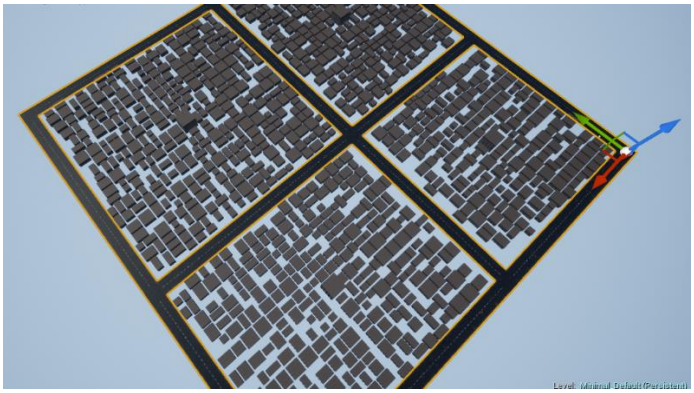


Рис.2. ─ Результат заповнення кварталів будинками

5. Генерація модульного будинку

В генерації будинку маємо такі параметри:
- Висота
- Ширина
- Довжина
- Розмір блоку будинку

Також користувач має можливість задати посилання на 3D-моделі (static mesh в контексті ігрового редактора Unreal Engine 4) як звичайне значення змінної. Таким чином він може вказати, які блоки треба використовувати для генерації фасаду будинку, які – для бокової частини, для даху, для задньої частини, для першого поверху.

Генерація самого будинку зводиться до створення потрійного циклу, який пройде по всім осям X Y Z (довжина, ширина і висота відповідно), де на кожній ітерації циклу буде знаходитись локальна координата (в локальному просторі актора-будинку на 3D сцені) для розміщення елемента будинку. Так, наприклад, маючи індекси ітерації циклу x, y, та z можна розрахувати координати блоків будинку в його локальному просторі.

Наприклад, маючи довжину 4, висоту 7 і ширину 5 отримаємо такий результат генерації:

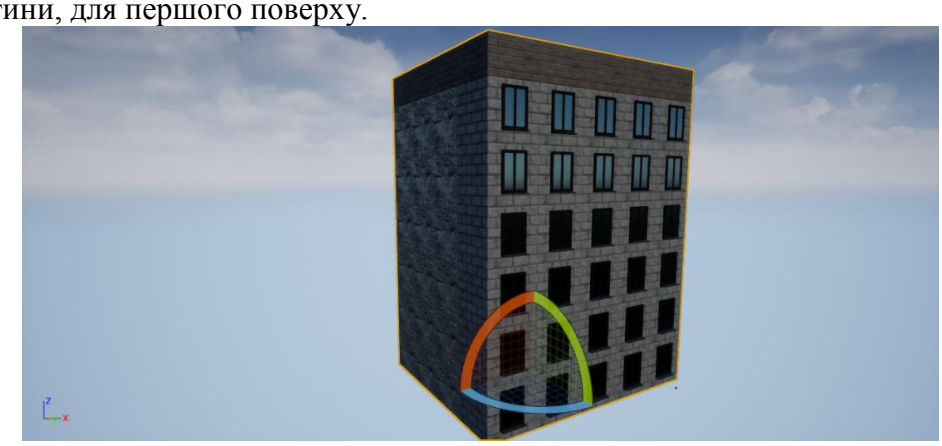


Рис.3. - Приклад згенерованого будинку

При цьому збоку, зверху, та попереду будинку були розміщені різні типи блоків.

6. Генерація парку

Алгоритм генерації парку основується на різновиді алгоритму комівояжера [4], коли для створення замкнутого шляху на масиві точок кожна точка з'єднується з найближчою вільною точкою.

Для початку візьмемо міський квартал (прямокутник) та розмістимо в ньому достатньо велику кількість точок (простим подвійним циклом по осям X та Y точки розміщуються на певній відстані одне від одного).

Наступний пункт теж доволі тривіальний – потрібно розмістити певну кількість точок, між якими буде проходити паркова доріжка. Для цього обираємо випадкову точку, видаляємо всі точки в певному радіусі від неї. Обираємо з точок що залишились ще одну точку і так далі, поки доступних точок не залишиться.

Отриманий масив точок з'єднується алгоритмом комівояжера та розміщуються 3D моделі паркової дороги.

Наступний етап – тривіальний, зводиться до розміщення декоративних елементів. На місці обраних паркових точок ставляться альтанки, вздовж доріг з певним інтервалом розміщуються лавки та ліхтарі, весь вільний простір парку заповнюється травою та деревами (дерева розміщуються хаотично, з невеликою ймовірністю в кожній точці, при умові що нове дерево не пересікається з вже розміщеним декоративним елементом). Оптимізація при розміщенні великої кількості моделей трави (та всіх інших моделей що повторюються в місті) забезпечується вбудованим в UE4 компонентом instanced static mesh [5]

7. Аналіз досліджень

В розробці ігрового рівня беруть участь три спеціаліста (або групи спеціалістів). За перший етап роботи відповідає гейм-дизайнер, який має створити вимоги до ігрового рівня, та описати його схематичний прототип. Задача левел-дизайнера – створити blockout (візуальне зображення ігрового рівня, де замість фінальних моделей виставляються прості блоки, які вказують на розміщення та розмір фінальних моделей).

Далі 3D environment artist (3D художник) має створити відповідні моделі, а левел-дизайнер – розставити їх на сцені (замінити blockout на фінальні моделі).

Даний процес значно спрощується, якщо розробники мають інструмент процедурної генерації карти. В такому випадку маємо лише декілька задач:

- Розміщення актора генерації та введення його налаштувань
- Створення 3D моделей, їх імпорт та передача їх до генератора
- Запуск генерації та отримання готового, згенерованого рівня

Генерація міста починається зі створення карти доріг, тому контролер генерації передає актору генерації доріг відповідне повідомлення. Після завершення генерації карти доріг контролер генерації отримає інформацію про зони в середині міста, які треба заповнити будинками або розмістити в них парки.

Генератор будинку розміщує власні блоки (моделі) відповідно до налаштувань, а генератор парку будує мережу доріг всередині себе, після чого створює актори-генератори паркових доріг та передає їм відповідні параметри. Кожен генератор паркової дороги розміщує дорогу та її декоративні елементи, після чого генератор парку заповнює весь вільний простір зеленими насадженнями.

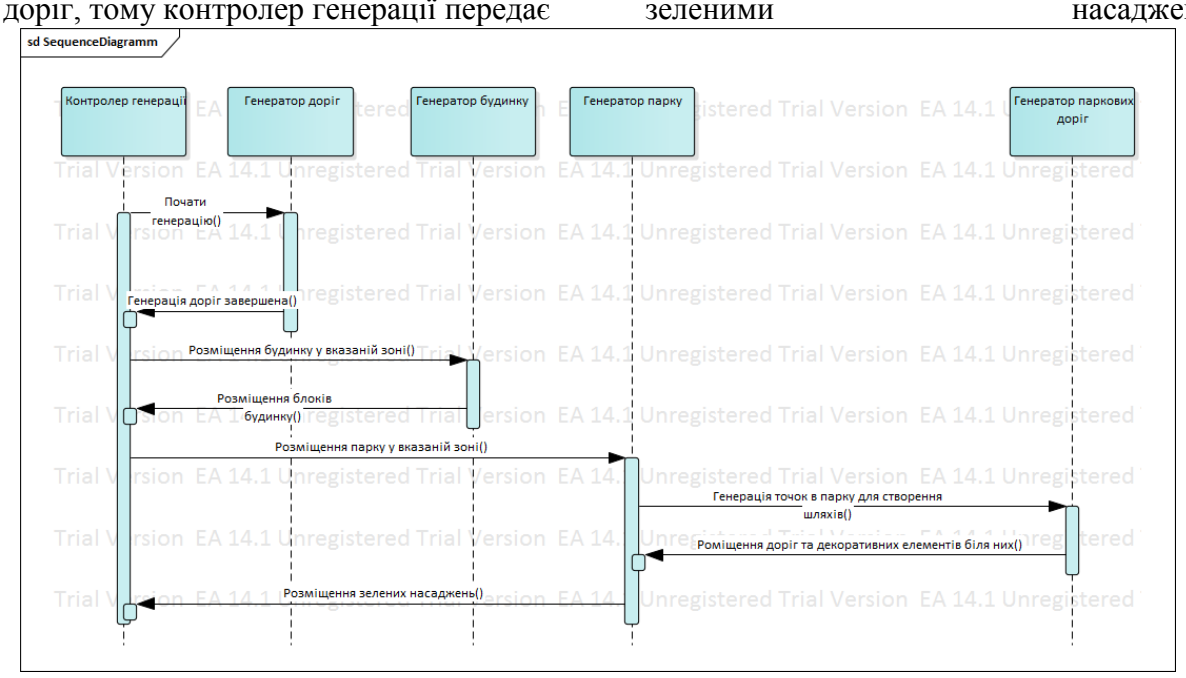


Рис.4. ─ Схема структурна послідовності. Послідовність генерації міста

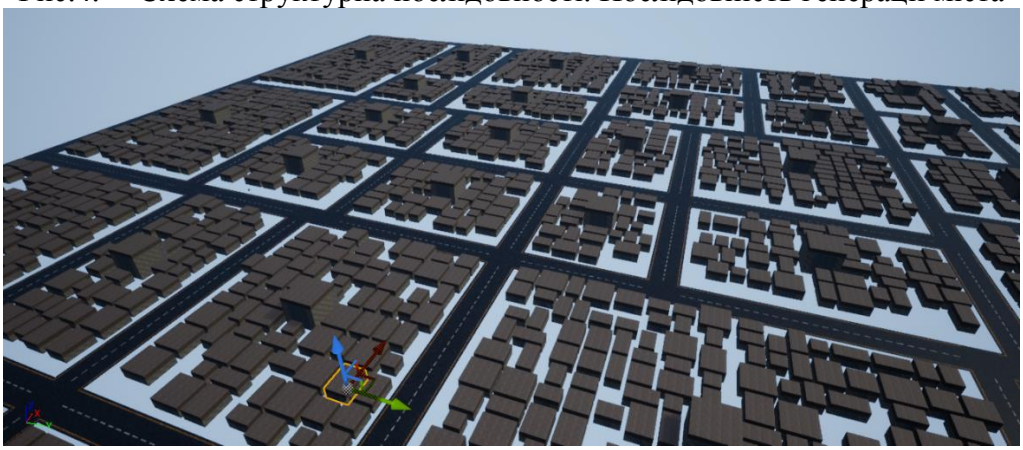


Рис.5. - Згенероване місто з розміщеними моделями доріг та будинками в кварталах

Рис.6. - Міський парк згенерований в кварталі

8. Вхідні та вихідні дані

Вхідні дані для системи процедурної генерації міста надходять від користувача системи (розробника, який використовує процедурну генерацію міста) у вигляді наступних налаштувань:

Налаштування системи генерації доріг:

- Алгоритм генерації доріг, який має бути обраний
- Мінімальний розмір міського кварталу;
- Максимальна довжина вулиць;
- Коефіцієнт відхилення дороги;
- Інтенсивність та типи декоративних елементів для дороги.

Налаштування системи генерації будинків:

- Мінімальна та максимальна висота, ширина та довжина будинків;
- Зміна розміру будинків від центру до околиць (задається функцією);
- Мінімальна дистанція між будинками;
- Відхилення розмірів будинків від норми.

Налаштування системи генерації «зеленої» місцевості:

- Актори, які треба розмістити (дерева, каміння, кущі і т.д.);
- Інтенсивність розміщення об’єктів;
- Кількість парків для генерації та їх налаштування.

На виході (після завершення генерації) створюється 3D сцена, яка одразу доступна для перегляду та редагування в Unreal Engine 4.

Висновки

В даній статті було обґрунтовано доцільність створення системи процедурної генерації міста з використанням засобів Unreal Engine 4. Були наведені математичні алгоритми, які використовуються в генерації, та описані результати їх роботи. Показано, що алгоритмічне забезпечення системи повинно включати в себе алгоритм діленням прямокутників, алгоритм генерації павутини, алгоритм розміщення обєктів в області та алгоритм комівояжера. Наведені результати процедурної генерації міста на 3D сцені.

Список літератури

1. Ammobox Studios. Procedural Building Lot [Електронний ресурс] / Ammobox Studios. – 2016. – Режим доступу до ресурсу: https://www.unrealengine.com/marketplace/en-US/slug/procedural-building-lot.

2. Andrey K. Procedural generator of the highways and roads [Електронний ресурс] / Andrey K. – 2018. – Режим доступу до ресурсу: https://www.unrealengine.com/marketplace/en-US/slug/procedural-generator-of-the-highways-and-roads?lang=en-US.

3. What is UE4? [Електронний ресурс] – Режим доступу до ресурсу: https://www.unrealengine.com/en-US/.

4. Задача коммивояжёра [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/.

5. UE4 ISM documentation [Електронний ресурс] – Режим доступу до ресурсу: https://api.unrealengine.com/INT/API/Runtime/Engine/Components/UInstancedStaticMeshComponent/index.html

УДК 519.854.2

ДУБИНА А.В.,
ЖДАНОВА Е.Г.,
МАРТЫНЮК Ю.Ю.,
СПЕРКАЧ М.О.

О ДОСТАТОЧНЫХ УСЛОВИЯХ ОПТИМАЛЬНОСТИ РАСПИСАНИЙ ВЫПОЛНЕНИЯ РАБОТ ПАРАЛЛЕЛЬНЫМИ ПРОПОРЦИОНАЛЬНЫМИ МАШИНАМИ

В работе представлены три задачи распределения заданного числа работ между пропорциональными машинами. Цель первой задачи состоит в минимизации общего времени выполнения всех работ. Во второй задаче необходимо максимизировать минимальное время завершения машинами своих работ. В третьей задаче целью является составление расписания с максимально равномерным распределением работ между машинами. В работе использован подход к решению этих задач, основанный на 1) преобразовании целевой функции как функции отклонения от нижней границы общего времени выполнения множества работ с заданным суммарным временем выполнения; 2) формулировании вспомогательных оптимизационных задач, которые позволяют установить нижние границы для исследуемых критериев; 3) решения вспомогательных оптимизационных задач определяют достаточные условия оптимальности исходных задач. Для сформулированных вспомогательных оптимизационных задач были разработаны алгоритмы их решения. Экспериментально установлено наиболее подходящее формальное описание критерия для третьей задачи.

КЛЮЧЕВЫЕ СЛОВА: РАСПИСАНИЕ, ПРОПОРЦИОНАЛЬНЫЕ ПАРАЛЛЕЛЬНЫЕ МАШИН, ДОСТАТОЧНЫЕ УСЛОВИЯ ОПТИМАЛЬНОСТИ, ВСПОМОГАТЕЛЬНАЯ ЗАДАЧА, КРИТЕРИЙ ОПТИМАЛЬНОСТИ

The paper presents three problems of distribution of a given number of jobs between proportional machines. The goal of the first task is to minimize the total execution time for all jobs. In the second task, it is necessary to maximize the minimum completion time of the machines by their machines. In the third task, the goal is to create a schedule with the most uniform distribution of work between machines. We used an approach to solving these problems based on 1) the transformation of the objective function as a function of the deviation from the lower limit of the total execution time of a set of jobs with a given total execution time; 2) the formulation of relevant auxiliary optimization problems that allow us to establish lower bounds for the criteria under study; 3) solutions of auxiliary optimization problems determine sufficient conditions for optimality of initial problems. For the formulated auxiliary optimization problems, algorithms for their solution were developed. The experimentally established the most appropriate formal description of the criterion for the third problem.

KEYWORDS: TIMETABLE, PROPORTIONAL PARALLEL MACHINES, SUFFICIENT CONDITIONS OF OPTIMALITY, AUXILIARY TASK, CRITERIA OF OPTIMALITY

Введение

Модели теории расписаний имеют широкое применение при планировании работ в разных сферах деятельности. Данная работа является продолжением исследования, представленного в работе [1], в которой рассмотрены три практически значимых задач теории расписаний, принадлежащих классу NP.

1 Постановка задачи

Задано множество работ $J = \{1,2,..,j,.,n\}$ и m параллельных машин различной скорости. Машины можно упорядочить по скорости выполнения работ и этот порядок одинаков для всех работ. Величина p_j является продолжительностью выполнения работы j на машине 1 (машина 1 считается эталонной). Скорость машины i обозначается через v_i, время, которое машина i тратит на выполнение работы j, равно p_j / v_i. Все работы множества J поступают одновременно в нулевой момент времени, процесс обслуживания каждой работы протекает без прерываний. Все машины начинают свою работу в нулевой момент времени.

Задача 1. Найти расписание, в котором достигает минимума максимальный момент завершения работ ($C_{\max}$). (в соответствии с принятой нотации задач теории расписаний, имеем задачу $Qm\|C_{\max}$ - минимизация makespan [2]).

Задача 2. Найти расписание, в котором достигает максимума минимальный среди моментов завершения работ машинами. (данная задача известна как *machine covering problem* [3].

Задача 3. Найти расписание с максимально равномерным распределением работ между машинами.

2 Результаты исследования задач

Первым шагом анализа задач является определение нижней границы C^* общего времени выполнения всех работ (нижней границы критерия оптимальности задачи 1) [3]: $C^* = P/V$, где $P = \sum_{j=1}^{n} p_j$, $V = \sum_{i=1}^{m} v_i$. Очевидно, что расписание, в котором машины завершают выполнение работ в момент времени C^* является оптимальным по всем анализируемым критериям.

Далее будем считать, что все p_j являются целыми числами, а машины упорядочены по скорости так, что $v_1 \ge v_2 \ge ... \ge v_m$. Введем величину $k_i = (v_i)^{-1}$ - коэффициент производительности машины i ($i = 1,...,m$), тогда выполняется $k_1 \le k_2 \le ... \le k_m$.

В работах [1,4-5] для анализируемых задач, было выполнено преобразование критериев оптимальности в функции отклонения от значения C^*. Это позволило сформулировать вспомогательные оптимизационные задачи, целью которых является определение идеальных контуров расписаний для соответствующих критериев.

2.1 Критерии оптимальности как функции отклонения от C^*

Пусть имеем некоторое расписание σ. Введем обозначение для

1) ***моментов завершения:*** $C_j(\sigma)$ - момент завершения работы j в расписании σ ; $S_i(\sigma) = \max_{j \in J_i(\sigma)} C_j(\sigma) = \sum_{j \in J_i(\sigma)} k_i p_j$ - момент завершения всех работ на машине i ($J_i(\sigma)$ - множество работ, выполняемых машиной i);

2) ***отклонений от C^*:***

$\Delta_i(\sigma) = \max\{0;\, S_i(\sigma) - C^*\}$, $R_i(\sigma) = \max\{0;\, C^* - S_i(\sigma)\}$

,

(выступ и резерв на машине i);

3) ***приведенных величин:***

$$c_i^* = C^* v_i = C^* / k_i ;$$

$$\Delta_i'(\sigma) = \max\{0; S_i'(\sigma) - c_i^*\};$$

$$R_i'(\sigma) = \max\{0; c_i^* - S_i'(\sigma)\}.$$

Задача 1. Критерий минимизации максимальной из длительностей загруженности машин (минимизации общего времени выполнения всех работ) имеет вид: $C_{\max} = \max_i S_i(\sigma) = C^* + \max_{1 \le i \le m} \Delta_i(\sigma) \to \min$. С учетом того, что $C^* = const$, критерий $C_{\max} \to \min$ сводится к минимизации максимального из выступов:

$$\max_{1 \le i \le m} \Delta_i(\sigma) \to \min \qquad (1)$$

Задача 2. Критерий максимизации минимального времени завершения работ имеет вид: $\min_i S_i(\sigma) = C^* - \max R_i(\sigma) \to \max$. Фактически этот критерий является симметричным критерию (1), задача сводится к минимизации максимального резерва:

$$\max_i R_i(\sigma) \to \min \qquad (2)$$

Задача 3. Расписанием с максимально равномерной загрузкой машин можно считать расписание, у которого

а) достигает минимума максимальное из отклонений моментов завершения машинами всех своих работ от идеального времени C^*, то есть расписание, в котором:

$$\max_i \{R_i(\sigma)\}; \{\Delta_i(\sigma)\} \to \min . \qquad (3)$$

б) достигает минимума разница между максимальными (слева и справа) отклонениями моментов завершения машин всех своих работ от C^*,то есть расписание, в котором:

$$\max_i \{R_i\} + \max_i \{\Delta_i\} \to \min . \qquad (4)$$

в) достигает минимума суммарное отклонение моментов завершения машин всех своих работ от C^*, то есть расписание, в котором:

$$\sum_{i=1}^{m} R_i + \sum_{i=1}^{m} \Delta_i \to \min . \qquad (5)$$

2.2 Достаточные условия оптимальности

С учетом вида критериев (1)-(5), расписание, в котором $\Delta_i(\sigma) = R_i(\sigma) = 0$, $i = 1,...,m$ является равномерным и оптимальным. Такое расписание можно получить только при условии целочисленности C^* и c_i^*, $i = 1,...,m$. Если для расписания σ выполняется: $S_i(\sigma) = k_i \sum_{j \in J_i} p_j = k_i c_i^* = C^*$, $i = 1,...,m$, то оно является оптимальным по всем критериям.

В работе рассматривается случай, когда $\exists i \mid c_i^* \notin Z$. Предположим, что мы имеем неполное расписание $\bar{\sigma}$, в котором на машины назначено некоторое подмножество $\overline{J} \subset J$ работ таким образом, что $\sum_{j \in \overline{J_i}} p_j = \lfloor c_i^* \rfloor$, где $\overline{J_i}$ – множество работ, которые в расписании $\bar{\sigma}$ выполняются машиной i, $i = 1,...,m$, $\cup \overline{J_i} = \overline{J}$. При этом остались нераспределенными работы с суммарной продолжительностью $\delta = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \sum_{j \in \overline{J_i}} p_j = . = \sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lfloor c_i^* \rfloor$. Это означает, что в неполном расписании $\bar{\sigma}$ на машины назначены работы с суммарной длительностью $P - \delta$. Предположим, что не распределенными осталось δ работ, каждая из которых имеет единичную длительность выполнения. С учетом критериев (1)-(5), перед нами встают следующие задачи: учитывая продуктивности машин распределить единичные работы в количестве δ между машинами таким образом, чтобы достичь минимума по этим критериям.

Вспомогательная оптимизационная задача 1. Математическая модель вспомогательной оптимизационной задачи (ВОЗ) для определения лучшего по критерию (1) распределения работ с общим объемом P имеет следующие составляющие.

Переменные. Определить x_i – количество работ единичной длительности, которые должны быть назначены на машину i, $i = 1,...,m$.

Ограничения. Общее количество работ составляет δ:

$$\sum_{i=1}^{m} x_i = \delta;$$

$$x_i \geq 0, \text{ целые, } i = 1,...,m.$$

Целевая функция. Минимизировать максимальный из выступов:

$$\max_i \{k_i (x_i - e_i)\} \to \min.$$

Математические модели других вспомогательных задач отличаются от модели ВОЗ 1 только целевыми функциями.

ВОЗ 2. Минимизировать максимальный из резервов:

$$\max_i \{k_i (e_i - x_i)\} \to \min.$$

ВОЗ 3. Максимально равномерно нагрузить машины.

Целевая функция задачи 3а:

$$\max_i \{ |k_i (e_i - x_i)| \} \to \min.$$

Целевая функция задачи 3б:

$$\max_i \{k_i (x_i - e_i)\} + \max_i \{k_i (e_i - x_i)\} \to \min.$$

Целевая функция задачи 3в:

$$\sum_{i=1}^{m} |k_i (e_i - x_i)| \to \min.$$

Алгоритмы решения ВОЗ

Входом всех алгоритмов являются: C^*, δ, e, k. На выходе имеем характеристики получаемых контуров расписаний, а именно: x - вектор распределения единичных работ, S^* - вектор моментов завершения машинами выполнения последних работ, $S^*_{\max}$ - момент завершения последней из работ (максимальная из продолжительностей занятости машин), $S^*_{\min}$ - минимальная из продолжительностей занятости машин.

ВОЗ 1. При назначении текущей работы, или увеличивается один из выступов, или появляется новый выступ. В зависимости от того, на какую машину распределяется работа, значения максимального выступа меняется по-разному.

Алгоритм 1 (алгоритм решения ВОЗ 1)

1 **for** $i := 1$ **to** m **do**

2 $x_i := 0$

3 $S_i^* := C^* - k_i e_i$

4 **end**

5 **while** $\delta \geq 1$ **do**

6 $q := \arg\left\{\min\left\{S_1^* + k_1; \ldots; S_m^* + k_m\right\}\right\}$

7 $x_q := x_q + 1$

8 $S_q^* := S_q^* + k_q$

9 $\delta := \delta - 1$

10 **end**

11 $S_{\max}^* := \max\left\{S_1^*, S_2^*, \ldots, S_m^*\right\}$

Величины $S_1^*, S_2^*, \ldots, S_m^*$ формируют идеальный контур расписания. Оптимальность полученного контура расписания доказана в [6].

ВОЗ 2. Схема алгоритма решения ВОЗ 2 отличается от *Алгоритма* 1 следующими пунктами: строку 6 заменяем на $q := \arg\left\{\max\left\{k_1 e_1; k_2 e_2; \ldots; k_m e_m\right\}\right\}$, строку 11 на $S_{\min}^* := \min\left\{S_1^*, S_2^*, \ldots, S_m^*\right\}$.

ВОЗ 3а. От *Алгоритма* 1 алгоритм решения этой задачи отличается следующим: после 3-й строки добавляем новые строки инициализации $\Delta_i' := 0$ - значение текущего приведенного выступа машины i; $R_i^* := k_i e_i$ - текущий резерв машины i; $\Delta_i^* := 0$ - текущий выступ машины i. Строку 6 заменяем на следующие строки:

$$q = \arg\left\{\min\left\{\begin{array}{l}\max\left\{k_1|x_1 - e_1 + \Delta_1'|; k_2|-e_2 + \Delta_2'|; \ldots; k_m|-e_m + \Delta_m'|\right\}; \\ \max\left\{k_1|-e_1 + \Delta_1'|; k_2|x_2 - e_2 + \Delta_2'|; \ldots; k_m|-e_m + \Delta_m'|\right\}; \\ \ldots \\ \max\left\{k_1|-e_1 + \Delta_1'|; k_2|-e_2 + \Delta_2'|; \ldots; k_m|x_m - e_m + \Delta_m'|\right\};\end{array}\right\}\right\};$$

$$\Delta_q' := x_q - e_q;$$

$$R_q^* := 0;$$

$$\Delta_q^* := k_q \Delta_q'.$$

А 11 строку заменяем следующими:

$$R_{\max}^* = \max\left\{R_1^*, R_2^*, \ldots, R_m^*\right\};$$

$$\Delta_{\max}^* = \max\left\{\Delta_1^*, \Delta_2^*, \ldots, \Delta_m^*\right\};$$

$$G_{\max}^* = \max\{R_{\max}^*; \Delta_{\max}^*\}.$$

ВОЗ 3б. Аналогично алгоритму решения ВОЗ 3а, изменяем инициализацию исходных переменных и изменяем строку 6 следующим образом:

$$q = \arg\left\{\min\left\{\begin{array}{l}\max\left\{k_1|e_1 - x_1 + \Delta_1'|; k_2|e_2 + \Delta_2'|; \ldots; k_m|e_m + \Delta_m'|\right\} \\ + \max\left\{k_1|x_1 - e_1 + \Delta_1'|; k_2|-e_2 + \Delta_2'|; \ldots; k_m|-e_m + \Delta_m'|\right\}; \\ \max\left\{k_1|e_1 + \Delta_1'|; k_2|e_2 - x_2 + \Delta_2'|; \ldots; k_m|e_m + \Delta_m'|\right\} \\ + \max\left\{k_1|-e_1 + \Delta_1'|; k_2|x_2 - e_2 + \Delta_2'|; \ldots; k_m|-e_m + \Delta_m'|\right\}; \\ \ldots \\ \max\left\{k_1|e_1 + \Delta_1'|; k_2|e_2 + \Delta_2'|; \ldots; k_m|e_m - x_m + \Delta_m'|\right\} \\ + \max\left\{k_1|-e_1 + \Delta_1'|; k_2|-e_2 + \Delta_2'|; \ldots; k_m|x_m - e_m + \Delta_m'|\right\};\end{array}\right\}\right\}$$

Результат работы (11-я строка) таков:

$$R_{\max}^* = \max\left\{R_1^*, R_2^*, \ldots, R_m^*\right\};$$

$$\Delta_{\max}^* = \max\left\{\Delta_1^*, \Delta_2^*, \ldots, \Delta_m^*\right\};$$

$$G_{\max}^* = R_{\max}^* + \Delta_{\max}^*.$$

ВОЗ 3в. Аналогично ВОЗ 3б, меняется 6-я строка:

$$q = \arg\left\{\min\left\{\begin{array}{l}\left\{k_1|x_1 - e_1 + \Delta_1'| + k_2|-e_2 + \Delta_2'| + \ldots + k_m|-e_m + \Delta_m'|\right\}; \\ \left\{k_1|-e_1 + \Delta_1'| + k_2|x_2 - e_2 + \Delta_2'| + \ldots + k_m|-e_m + \Delta_m'|\right\}; \\ \ldots \\ \left\{k_1|-e_1 + \Delta_1'| + k_2|-e_2 + \Delta_2'| + \ldots + k_m|x_m - e_m + \Delta_m'|\right\};\end{array}\right\}\right\},$$

а также последние строки, определяющие результирующее значение критерия:

$$R^* = \sum_{i=1}^{m} R_i \; ; \Delta^* = \sum_{i=1}^{m} \Delta_i \; ; \; G^* = R^* + \Delta^*.$$

3 Результаты экспериментов

Был проведен сравнительный анализ оптимальных решений сформулированных ВОЗ для двух классов индивидуальных задач, которые обозначены как *S* и *M* (в классе *S* значения производительностей машин были равномерно распределены в интервале [1÷1.2], в классе *M* - [1÷6]); для каждого класса рассматривались такие количества машин: $m = 3$, 4, 5, 10, 15 ..., 35; для каждого значения m было сгенерировано по 100 индивидуальных задач с длительностями работ из интервала [13÷98].

На первом этапе экспериментов сравнивались контуры, получаемые на основе решений ВОЗ 3а, 3б и 3в. Рисунок 1 иллюстрирует влияние количества машин на процент совпадения оптимальных контуров расписаний по критериям 3а, 3б и 3в.. Как видно из представленных графиков, контуры расписаний задач класса *M* совпадают чаще, нежели у *S*. Это объясняется тем, что в классе *S* продуктивности машин находятся в более узком интервале. Для задач класса *M* продуктивности машин значительно отличаются друг от друга, это влияет на решения ВОЗ: построенные контуры совпадают чаще. При небольшом количестве

машин (до 10) процент совпадения результатов решения задач составляет более 15 (для задач класса *M*) и более 5 (для задач класса *S*). С увеличением числа машин - процент совпадения уменьшается.

Анализ результатов показал, что критерий (4) ВОЗ 3б наиболее соответствует формулировке целевой функции задачи 3.

На втором этапе сравнивались контуры полученных с решения ВОЗ 1, 2, 3 (3б). Эксперименты показали, что контуры задач класса *M* совпадают чаще, чем у задач класса *S*. При увеличении количества машин – процент совпадения уменьшается. Результаты (влияние количества машин на процент совпадения оптимальных контуров расписаний) представлены на рис. 2.

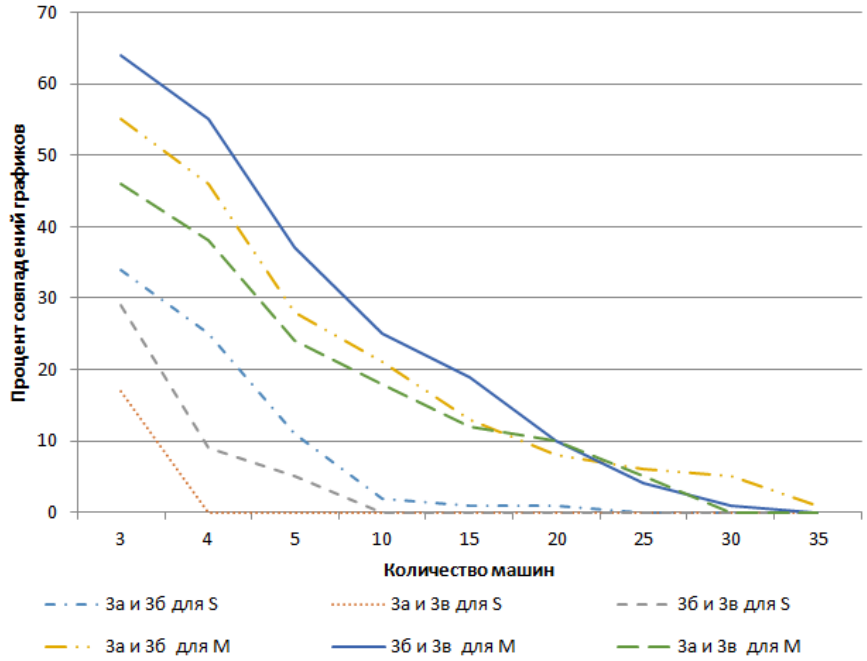


Рис. 1 – Влияние количества машин на процент совпадения оптимальных контуров расписаний по критериям 3а, 3б и 3в.

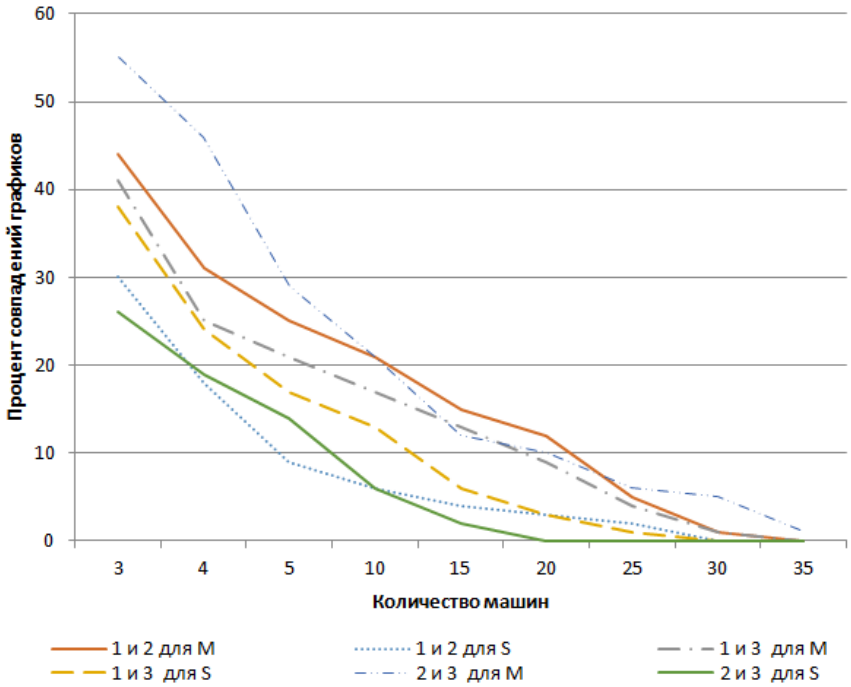


Рис. 2 – Влияние количества машин на процент совпадения оптимальных контуров расписаний по критериям 1, 2 и 3

Выводы

В работе было экспериментально исследован подход к решению трех близких задач теории расписаний для пропорциональных машин, основанный на преобразовании целевой функции в функцию отклонения от нижней границы общего времени выполнения множества работ с заданным суммарным временем выполнения и формулировании соответствующих вспомогательных оптимизационных задач. Для пяти сформулированных вспомогательных оптимизационных задач были разработаны алгоритмы их решения, которые по сути являются жадными, но при этом позволяют находить оптимальные решения.

Решения вспомогательных оптимизационных задач (контуры оптимальных расписаний) определяют достаточные условия оптимальности исходных задач.

В результате серии экспериментов установлено наиболее подходящее формальное описание критерия для задачи 3; определено влияние разброса продуктивностей машин и их количества на процент совпадения оптимальных контуров расписаний для исследуемых критериев.

СПИСОК ЛИТЕРАТУРЫ

1. Popenko V., Sperkach M., Zhdanova O., Kokosinski Z.: On Optimality Conditions for Job Scheduling on Uniform Parallel Machines. In: Hu Z., Petoukhov S., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing, vol. 938, pp. 103-112. Springer, Cham (2020) // The Second International Conference on Computer Science, Engineering and Education Applications ICCSEEA 2019, 26–27 January 2019, Kiev, Ukraine. – 2019, pp. 103-112. doi: 10.1007/978-3-030-16621-2_10

2. Pinedo M.L. Scheduling. Theory, Algorithms and Systems / Michael L. Pinedo. – Springer, 2008.– 671 p.

3. Senthilkumar P. Literature Review of Single Machine Scheduling Problem with Uniform Parallel Machines / P. Senthilkumar, S. Narayanan // Intelligent Information Management. – 2010. – №2. – P. 457–474.

4. Сперкач М.О. Задача визначення максимально пізнього моменту початку виконання завдань із спільним жорстким директивним терміном паралельними пристроями різної продуктивності / М.О. Сперкач // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2015. – №63 – С.12-18.

5. Сперкач М.О. Складання розкладу виконання завдань паралельними пристроями різної продуктивності з метою максимально рівномірного завантаження пристроїв / М.О. Сперкач, О.Г. Жданова // Вісник ХНТ ім. В.Н. Каразіна. Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». – Х.: 2015. – №1156 – С.92-106.

6. Liao C.J. Makespan Minimization for Multiple Uniform Machines / C.J. Liao, C.H. Lin // Computers & Industrial Engineering. – 2008 – v. 54. – n. 4. – P. 983-992.

УДК 004.912

ДВОРНИК В. А.,
КОВАЛЮК Т. В.

ПОБУДОВА ІНДИВІДУАЛЬНИХ ОСВІТНІХ ТРАЄКТОРІЙ ТА ВИЗНАЧЕННЯ ВМОТИВОВАНОСТІ СТУДЕНТІВ НА ОСНОВІ МЕТОДУ ЛАТЕНТНО-СЕМАНТИЧНОГО АНАЛІЗУ

В даній статті розглянуто практичне застосування методу латентно-семантичного аналізу на прикладі задачі визначення вмотивованості студентів. Розглянуто проблему перевірки вмотивованості студентів під час вибору спеціальності та побудови індивідуальної освітньої траєкторії. Приведено аналіз ефективності паралельних обчислень алгоритму, визначено основні показники якості – прискорення і ефективність. Наведено графіки залежностей виміряних величин від кількості процесів.

КЛЮЧОВІ СЛОВА: ІНДИВІДУАЛЬНА ТРАЄКТОРІЯ, ВМОТИВОВАНІСТЬ, ЛАТЕНТНО-СЕМАНТИЧНИЙ АНАЛІЗ, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, ПРИСКОРЕННЯ, ЕФЕКТИВНІСТЬ.

In this article considered the practical using of the latent semantic analysis method on the example of the task of determining the motivation of students. The problem of checking students' motivation in choosing a specialty and constructing an individual educational trajectory is reviewed. The analysis of the efficiency of parallel algorithm calculations is given, the main quality indicators are defined – acceleration and efficiency. The graphs of the dependencies of the measured values on the number of processes are shown.

KEYWORDS: INDIVIDUAL TRAJECTORY, MOTIVATION, LATENT-SEMANTIC ANALYSIS, PARALLEL CALCULATIONS, ACCURACY, EFFICIENCY.

1. Вступ

Перевиробництво непотрібних фахівців, підготовка недостатньо кваліфікованих працівників, низька частка зайнятості випускників – все це проблеми, закорінені в неправильному виборі професії. Велика кількість школярів погано орієнтуються в світі професій, недостатньо знають вимоги до майбутньої роботи і погано їх співвідносять зі своїми здібностями. Автоматизація визначення ступеня мотивації студентів при виборі майбутньої спеціальності та об'єктивного співвідношення вимог до майбутньої професії зі своїми здібностями є актуальною задачею через експоненціальне зростання обсягу інформації, яку слід проаналізувати, значну долю невизначеності при виборі освітніх траєкторій, складний мотиваційний процес і низьку якість точності пошуку інформації.

2. Постановка задачі

Розглянемо проблему перевірки вмотивованості студентів під час вибору спеціальності та побудови індивідуальної освітньої траєкторії. В якості вхідної інформації розглядатиме мотиваційні листи, есе та тести з професійної орієнтації з вільними відкритими відповідями. Для фільтрації, рубрикації і кластеризації документів, пошук відповідей на запитання, автоматичне анотування документів, пошук схожих документів і дублікатів, автоматизованої оцінки якості вільних розгорнутих відповідей пропонується залучити метод латентно-семантичного аналізу (LSA) [1]. Цей метод обробки інформації природною мовою аналізує взаємозв'язок між колекцією документів і термінами, що в них зустрічаються, зіставляє тематики всім документам і термам.

3. Метод латентно-семантичного аналізу

Для застосування методу латентно-семантичного аналізу потрібно побудувати $m \times n$ матрицю X «термін-документ», з комірками x_{ij}, що містять вагові коефіцієнти терміну t_i, в документі d_j. Стовпці матриці X на практиці відповідають мультимножині слів для документа, при цьому можуть бути використані терміни, зважені за якою-небудь схемою, наприклад, за схемою TF-IDF [2]. Отримана матриця «термін-документ» X являє собою просторово-векторну модель подання текстової інформації і є вхідними даними для методу латентно-семантичного аналізу та методу латентного розміщення Діріхле [3]. Потрібно визначити, чи відповідає вхідний текст тематиці певної професійної сфери та визначити тему цього тексту, застосовуючи методи семантичного аналізу тексту.

Після побудови матриці X «термін-документ» здійснюється її сингулярне розкладання (SVD) на три матриці: $X = USV^t$, де U, V^t – ортогональні матриці, S – діагональна матриця, яка містить власні значення, упорядковані за спаданням. Кожне з власних значень відповідає одному з компонентів, що відслідковуються в колекції документів, і позначає, наскільки цей компонент важливий у всій колекції. Кількість рядків матриці X зменшують, зберігаючи при цьому структуру подібності у стовпцях. Потім терми/документи порівнюють за допомогою обчислення косинуса кута між

двома векторами (скалярний добуток векторів, поділений на добуток їх модулів), що утворений будь-якими двома рядками. Значення, близькі до 1, означають схожість термів/документів, тоді як значення, близькі до 0, представляють їх різнорідність.

Для визначення приналежності заданого тексту певній тематиці застосовується латентне розміщення Діріхле, згідно з яким кожний документ розглядається як набір різних тем. Розподіл тем має розподіл Діріхле [3]:

$$f(x_1, \dots, x_k; a_1, \dots a_k) = \left(\frac{1}{\beta(a)}\right) \prod_{i=1}^{k} x_i^{a_i - 1};$$

$$\sum_{i=1}^{k} x_i = 1;$$

$$a_i > 0.$$

Для зменшення обчислювальної складності алгоритму та підвищення швидкодії деякі кроки алгоритму можуть бути розбиті на паралельні задачі і виконані паралельно. Для цього створюється словник термінів, що стосуються певної професійної сфери. Терміни перевіряються на відповідність вхідному тексту. Перевірка відповідності здійснюється паралельно різними процесами.

4. Дослідження паралельних властивостей алгоритму

На рисунку 1 зображена ярусно-паралельна форма алгоритму. Алгоритм в ярусно-паралельній формі представляється у вигляді ярусів, причому в нульовий ярус входять оператори (гілки) незалежні один від одного.

На графі позначені переходи, які означають передачу результатів обчислення примітивної операції з одного ярусу до операції з наступного ярусу. Це можна назвати викликами методів в алгоритмі, в які передаються певні дані, які були результатами методів, які виконувались перед цим. Яруси діляться по переходах.

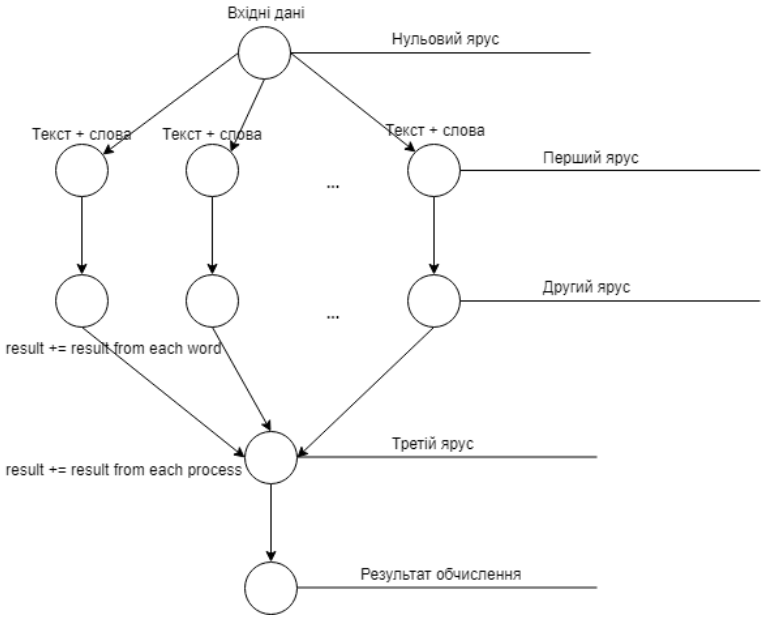


Рис. 1. Ярусно-паралельна форма алгоритму

5. Дослідження ефективності паралельних обчислень алгоритму

Для оцінки оптимальності розроблюваних методів паралельних обчислень наведені широко використовувані в теорії і практиці паралельного програмування основні показники якості – прискорення (speedup), що показує, у скільки разів швидше здійснюється рішення завдань при використанні декількох процесорів, і ефективність (efficiency), яка характеризує частку часу реального використання процесорів обчислювальної системи.

Складність алгоритму латентно-семантичного аналізу дорівнює: $O(|V|Mk^2)$, де $|V| \times M$ – розмірність матриці «термін-

документ», а k – залишені рядки з матриць сингулярного розкладання.

Можемо вважати, що для вирішення задачі послідовно потрібно часу $T_1 = |V|Mk^2$, а для вирішення паралельно $T_p = \frac{|V|Mk^2}{p}$. Обчислимо прискорення:

$$S_p = \frac{T_1}{T_p} = \frac{|V|Mk^2}{\frac{|V|Mk^2}{p}} = p$$

Тепер обрахуємо прискорення роботи програми за формулою Амдала:

$$S_p = \frac{1}{f + \frac{1-f}{p}}$$

Враховуючи те, що всі операції, що виконується в рамках алгоритму розпаралелені, частка коду, яка не може бути розпаралелена (f) дорівнює 0, тобто

$$S_p = \frac{1}{0 + \frac{1-0}{p}} = p$$

Ефективність використання процесорів при паралельній реалізації алгоритму оцінюється величиною:

$$E_p(n) = \frac{T_1}{pT_n} = \frac{S_p}{p} = \frac{p}{p} = 1$$

Отже, теоретично, розпаралелювання алгоритму латентно-семантичного аналізу є доцільним, адже досягається прискорення роботи алгоритму.

На рисунку 2 зображено залежність прискорення алгоритму від кількості процесів (дослідження проводились на тексті з кількість слів – 559), на рисунку 3 – ефективності використання процесорів від кількості процесів.

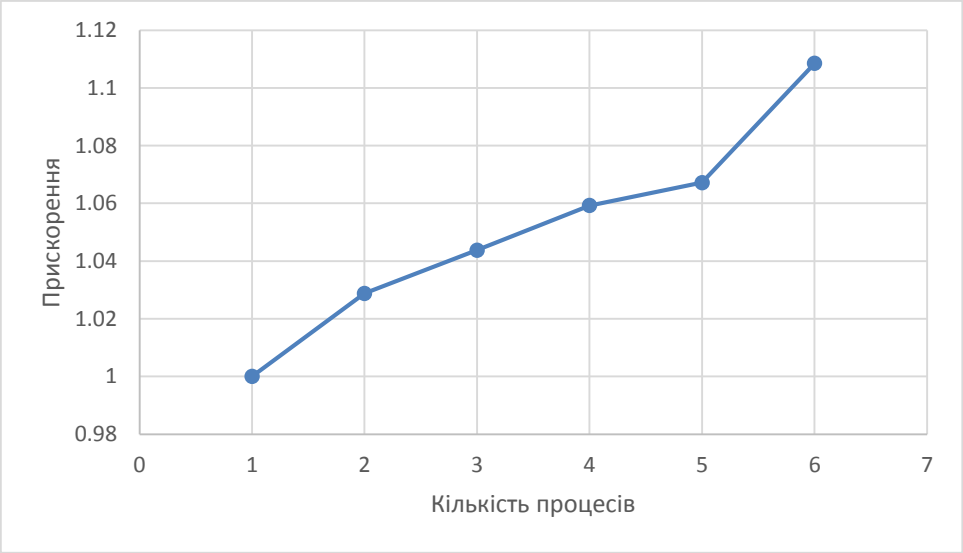


Рис. 2. Залежність прискорення від кількості процесів

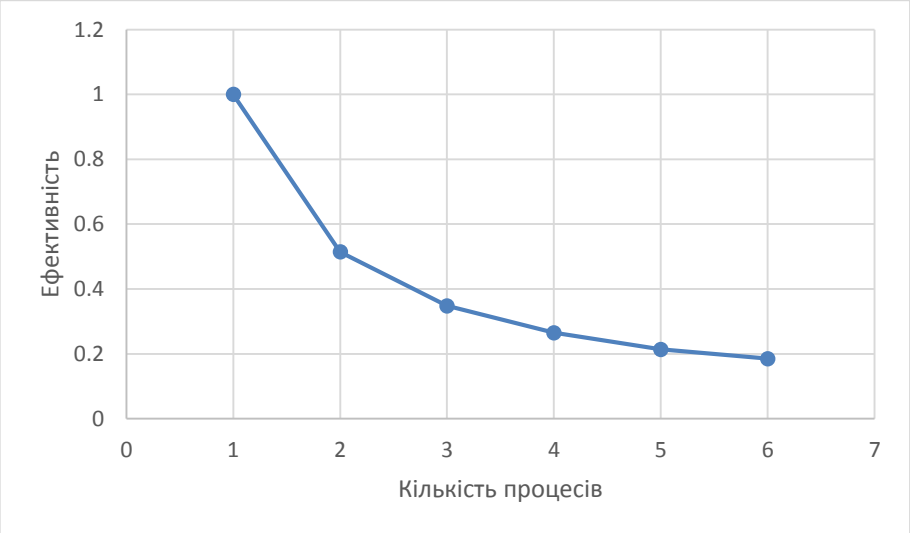


Рис. 3. Залежність ефективності від кількості процесів

З рисунків видно, що прискорення зі збільшенням кількості процесорів зростає, проте значення дещо менші за теоретичні, але це не критично, а графік залежності ефективності від кількості процесів спадає, через те що, як можна було помітити, прискорення не зростає різко і залишається більш менш на одному рівні.

Висновки

Застосування методу латентно-семантичного аналізу і латентного розміщення Діріхле до аналізу мотиваційних листів студентів і результатів тестування студентів з відкритими відповідями щодо їх професійного спрямування дає можливість визначити ступінь мотивації студентів при виборі спеціальності, їх можливість опановувати вибрані дисципліни, а значить, будувати індивідуальні траєкторії навчання.

У дослідженнях паралельності алгоритму було виявлено, що прискорення алгоритму збільшується із кількістю процесорів, а ефективність використання процесорів зменшується при збільшенні кількості значень. Усі отримані практичні оцінки не перевищують обраховані теоретичні значення.

Список використаних джерел

1. Застосування методу латентно-семантичного аналізу для автоматичної рубрикації документів. [Електронний ресурс]. Режим доступу: https://cyberleninka.ru/article/n/primenenie-metoda-latentno-semanticheskogo-analiza-dlya-avtomaticheskoy-rubrikatsii-dokumentov
2. TF-IDF. [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/ wiki/TF-IDF
3. Nicholas Ruozzi. Latent Dirichlet Allocation [Електронний ресурс]. Режим доступу:https://www.utdallas.edu/~nrr150130/cs6375/2015fa/lects/Lecture_20_LDA.pdf

УДК 004.93(015.7)

САЛИЙ О.А.

РАСПОЗНАВАНИЕ ЭМОЦИЙ ЧЕЛОВЕКА В РЕЖИМЕ ОНЛАЙН

В данной статье расскрыта актуальность темы автоматизации распознавания эмоций человека, а также приведены примеры использования данного решения в повседневной человеческой жизни. Проанализированы существующие методы решений данной задачи. Предложен главный алгоритм по автоматизации процесса распознавания эмоций человека с примерами используемых алгоритмов и технологий.

РАСПОЗНАВАНИЕ ЛИЦА, РАСПОЗНАВАНИЕ ЭМОЦИЙ, МАШИННОЕ ОБУЧЕНИЕ, НЕЙРОННЫЕ СЕТИ, ЭМОЦИИ

This work is aimed at modern developments in the field of computer vision and the problem of human’s emotions recognition in real time. The most popular emotions recognition methods are considered. The basic principles of emotions classifications are considered. The main algorithm of face recognition is proposed. Reviewed main ideas of convolutional neural networks (CNN), their advantaged, and efficient approach to their training with big data sets.

FACE RECOGNITION, EMOTION RECOGNITION, MACHINE LEARNING, NEURAL NETWORKS, EMOTIONS

Введение

Полноценное общение между людьми невозможно без проявления и анализа эмоций. Поэтому при создании современных человеко-машинных систем актуально применение методов автоматического распознавания эмоций.

Одним из основных способов распознаваемых эмоций человека другим человеком является анализ визуальной информации. Поэтому автоматизация этого

процесса очевидно должна быть основана на использовании методов и средств компьютерного зрения.

Компьютерное зрение является научной областью, в рамках которой ведутся исследования по изучению теории и фундаментальных алгоритмов анализа изображений объектов и сцен. Задача распознавания эмоций может использоваться в системах, применяемых в различных сферах человеческой деятельности. Рассмотрим несколько из них.

В настоящее время активно развивается робототехника. При этом следует отметить, что основное направление научных и практических работ связан с созданием интеллектуальных роботов. Ранее основной сферой применения робототехнических систем была промышленность. С развитием новых технологий появились бытовые работы, примером которых может служить робот-пылесос, оснащенный средствами машинного зрения. Машина зрение в робототехнических системах предназначеный в первую очередь для определения текущего положения, анализа окружающей обстановки, объезда препятствий, выявление заданных предметов и т.д. Распознавание эмоций человека бытовым роботом естественным образом позволяет повысить уровень интеллектуализации их взаимодействия, например, для обеспечения правильного реагирования на состояние человека.

Особенно актуально правильно определить состояние человека в случаях, связанных с опасностью для его жизни. В качестве одного из примеров можно привести системы распознавания усталости человека, которыми оснащаются некоторые современные автомобили. Подобные системы позволяют в ряде случаев избежать аварий, вызванных невнимательностью, сонливостью или плохим самочувствием водителя. Анализ осуществляется на основе результатов обработки изображений человека, полученных с видеокамеры.

Еще одной сферой применения методов автоматического распознавания эмоций является обеспечение безопасности людей с помощью автоматизированных охранных систем. Современные охранные системы часто имеют в своем составе средства регистрации и анализа видеоданных. Однако, как правило, в таких системах решение принимается человеком-оператором. Это может привести к снижению точности и оперативности реагирования на различные ситуации, связанные с поведением людей на охраняемых территориях. Повысить эффективность охранных систем можно за счет повышения степени автоматизации процедур, обеспечивающих анализ изображений. Это позволяет сделать вывод об актуальности создания интеллектуальных детекторов для анализа нештатных ситуаций. Среди других важных задач, решение которых необходимо обеспечить в конкретных охранных системах, можно выделить обнаружение и сопровождение людей, которые проявляют эмоции, характерные для нарушителей правопорядка, психически больных, террористов и т.д.

Методы обработки и анализа изображений в интеллектуальных детекторах внештатных ситуаций развиваются в рамках видеоаналитики. К данному направлению относятся также методы, которые используются в маркетинговых системах.

Существующие методы

Существует несколько методов распознавания эмоций:

1. Система кодирования лицевых движений (Facial Action Coding System (FACS))

Представляет собой систему для классификации выражений лица человека, изначально разработанную Полом Экманом и Уоллесом Фризеном в 1978 году

2. Скрытая марковская модель (Hidden Markov Model)

Статистическая модель, имитирующая работу процесса, похожего на марковский процесс с неизвестными параметрами, и задачей ставится разгадывание неизвестных параметров на основе наблюдаемых. Полученные параметры могут быть использованы в дальнейшем анализе, например, для распознавания образов. СММ может быть рассмотрена как простейшая байесовская сеть доверия.

3. Eigenfaces

На основе анализа существующих методов распознавания лиц человека представлена

новая методика выделения изображения. И в сочетании с искусственной нейронной сетью появился новый метод распознавания человеческого лица. Благодаря извлечению алгебраической характеристики образца, собственных значений изображения человеческого лица, классификатор нейронной сети обучается распознаванию. В основе алгоритма лежит использование фундаментальных статистических характеристик: средних (мат. Ожидание) и ковариационной матрицы; использование параметра главного компонента.

4. Нейронные сети (Neural Networks)

Распознавание лица как идея происходит от реального термина «распознавание лица». Часть человеческого "распознавания лица" - это "память" людей, что означает мозг. Связь между этим термином биологической организации и компьютера заключается в нейронных сетях. Нейронная сеть представляет собой взаимосвязанную группу узлов, сродни обширной сети нейронов человеческого мозга.

В силу популярности нейронных сетей в мире информационный технологий, в данной работе будет рассматриваться и использоваться именно этот метод.

Алгоритм

Пол Экман (исследователь и автор самых известных работ, посвященных исследованию невербального поведения) в своих работах выделяет семь базовых эмоций [1]: радость, удивление, печаль, гнев, отвращение, презрение и страх. Основываясь на этих данных, было принято решение разделить все анализируемые изображения на 7 классов. Для обучения модели был выбран заранее подготовленный набор из 28000 изображений, по 4000 объектов, относящихся к каждому классу. При разработке сервиса, который отвечает за распознавание эмоций, использовался язык программирования Python. На данный момент этот язык является одним из наиболее популярных в области машинного обучения. Применялись библиотеки Tensorflow и Keras, которые продоставляют удобный высокоуровневый интерфейс для работы с нейронными сетями.

Как показала практика - лучше всего для разпознавания и классификации изображений подходят сверточные нейронные сети (рис. 1). Такие сети позволяют обрабатывать изображения не полностью, а отдельными блоками, постепенно уменьшая их размеры и выделяя наиболее важные признаки. Это признаки определяются автоматически в ходе обучения, при этом наиболее явные являются крайне устойчивыми к посторонним шумам [2].

Основная идея последовательной разработки программы и процесса распознавания (рис. 2) делится на следующие этапы:

1. Определить лицо.

Используем классификатор, основанный на характерных признаках Хаара и алгоритме обучения AdaBoost, для распознавания лица. Этот метод распознавания впервые был предложен Paul Viola и улучшен Rainer Lienhart [3]. Классификатор состоит из каскадных подклассификаторов.

2. Найти органы.

В рассматриваемом методе использовался тот же классификатор Хаара, как в случае распознавания лица, для получения информации об органах. Чтобы ускорить распознавание и уменьшать ложные обнаружения, на лице были заданы определейнные зоны (как представлено на рисунке 2). Идея простая: рот всегда находится в нижней части лица, брови и глаза всегда сверху рта, поэтому программе не надо искать объект в другом месте изображения, таким образом, возможные ложные распознавания в неверном месте экранируются и время распознавания уменьшается.

3. Ключевые точки.

Принцип получения этих точек заключается в следующем: перейти от цветного изображения к полутоновому, затем к бинарному, и последнее: получить ключевые точки (рис. 3). Переход от цветного изображения до полутонового выполняется обычно на шаге захвата и отслеживания лица. Для получения бинарного изображения был использован адаптивный порог, который может уменьшать отрицательный эффект влияния перепадов света и тени на получаемом от видеокамеры цифровом изображении. Затем чтобы получить точки полученное изображение обрабатывалось с

помощью градиентной маски и выделении средних и концевых точек (рис. 3).

4. Классификация.

Распознаются 6 общих эмоций - удивление, страх, отвращение. гнев, счастье, грусть, характеристики которых представлены в таблице на рисунке 4.

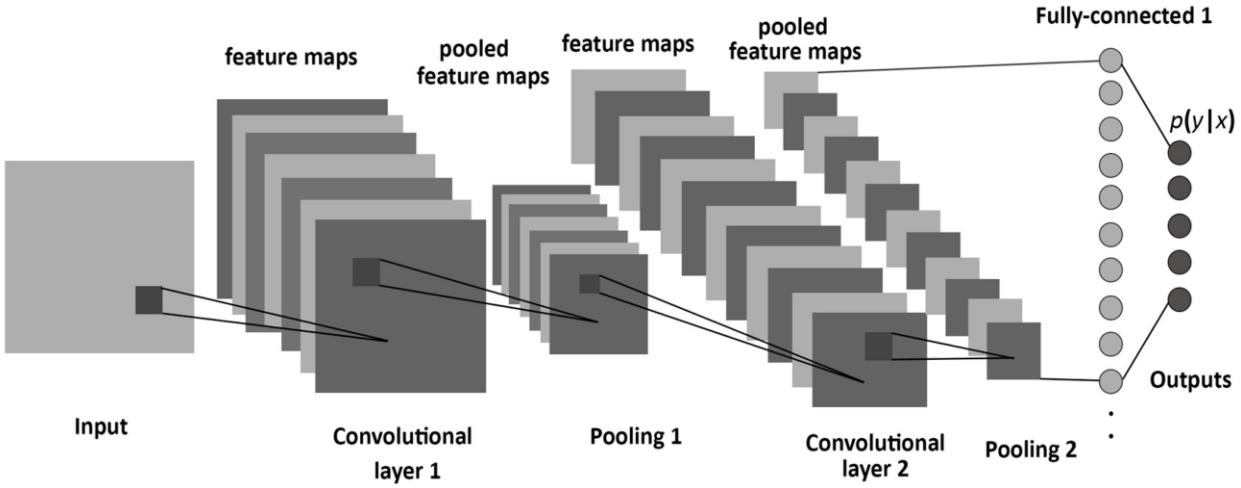


Рис. 1 - Пример сверточной нейронной сети

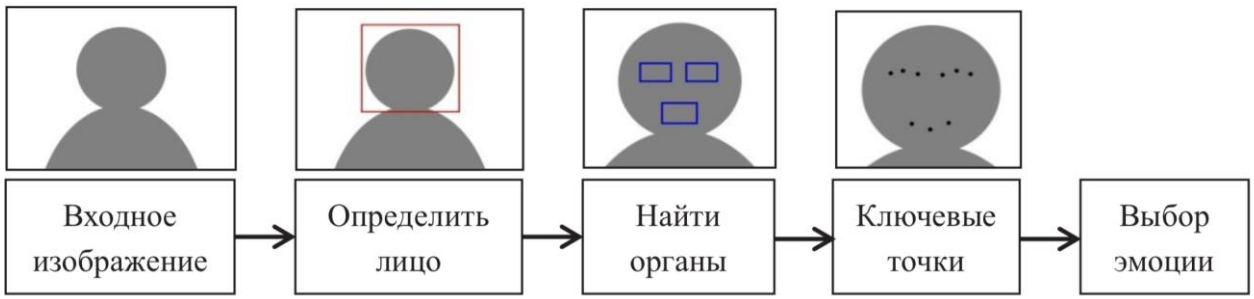


Рис. 2 - Процесс распознавания

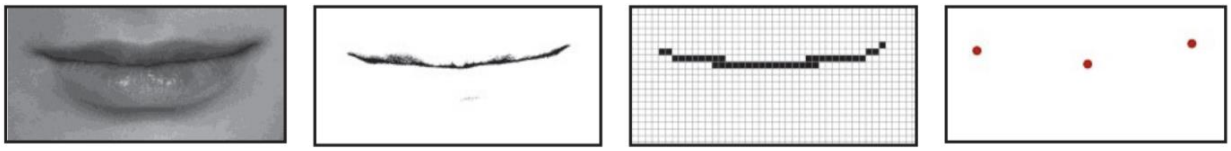


Рис. 3 – Определение ключевых точек

Эмоция	Бровь	Рот
Удивление	Поднимется	Открывается
Страх	Поднимется и сморщится	Открывается и растягивается
Отвращение	Снизится	Поднимутся и конечники снижуться
Гнев	Снизится и сморщится	Открывается или конечники снижуться
Счастливый	Пригибается	Конечники поднимутся
Грусть	Конечник снизится	Конечники снижуться

Рис. 4 – Характеристики форми органов

Заключение

В данной работе была проанализирована актуальность темы распознавания эмоций, а также приведены возможные примеры использования рассматриваемой темы. Были рассмотрены широко используемые методы решения этой задачи.

В работе рассмотрен простой метод для автоматического определения эмоций на цифровых изображениях. В процессе обнаружения применялся классификатор Хаара и алгоритмы цифровой обработки изображений. Метод имеет приемлемый уровень обнаружения. Этот метод может быть использован в интерфейсе интеллектуальной системы для создания условий естественного и интуитивного человеко-машинного взаимодействия.

Список литературы

1. Tim Dagleish, Mick J. Power Handbook of Cognition and Emotion. / T.Dagleish, M. Power - Sessux, U.K.: John Wiley & Sons, Ltd., 1999. - 47p.
2. Arriaga O. Real-time Convolutional Neural Networks for Emotion and Gender Classification. / Octavio Arriaga, Matias Valdenegro-Toro, Paul Ploger // CoRR. - 2017.
3. Viola P., Jones M. Robust Real Time Object Detection // 8th IEEE International Conference on Computer Vision. Vancouver, 2001, pp. 151 - 155.

УДК 004.021

КОСЯК О.М.
ПРОСКУРА С.Л.

КЛІЄНТ-СЕРВЕРНА АНАЛІТИЧНА ПРОГРАМА УПРАВЛІННЯ РОБОЧИМ ЧАСОМ

В даній статті, розглянута клієнт-серверна аналітична програма з оптимізації та управлінням використання робочого часу. Впровадження цієї системи дозволить оптимізувати робочий час при ненормованому робочому графіку, та знизити шкідливі впливи на здоров’я пов’язані з довгим сидінням в одній позі, та напруженням зору від довгого перебування за комп’ютером. Також це дозволить створювати кілька програмно-згенерованих розпорядків за короткий час.

КЛЮЧОВІ СЛОВА: Клієнт-серверна, аналітична програма, використання робочого часу, робочий графік, програмно-згенеровані розпорядки.

In this article, the client-server analytical system for optimizing and managing the use of working time is considered. The introduction of this system will optimize working hours with non-standardized work schedules, and reduce harmful health effects associated with long seating in one posture, and the strain of vision from long stay at a computer. It will also allow you to create several program-generated schedules in a short time.

Вступ

Зараз на ринку представлена велика кількість програм, так званих «Планувальників», вони відрізняються зазвичай лише дизайном, а функціонал у них в більшості однаковий: користувач сам, вручну планує свої дії. При плануванні свого робочого часу людина може спиратись лише на себе і свій досвід. В нашу задачу входить: допомогти користувачу розпланувати свій час так, щоб він максимально ефективно виконував свою роботу (на основі різних даних будуть надані автоматичні рекомендації для поліпшення існуючого розпорядку, або згенерований оптимальний розпорядок «з нуля», на основі даних введених користувачем), при цьому мінімізувати шкідливий вплив на його організм (через постійне сидіння в одному положенні, перевтому, порушення режиму харчування) що відповідно знизить рівень уваги та ефективність праці..

На даний момент існує певна кількість аналогів, які вирішують подібні задачі. Система для управління проектами YouTrack [1], повноцінний планувальник LeaderTask [2], і подібні аналоги спрощують створення та

генерацію розпорядку. Проте немає відомого аналогу, який би автоматично генерував розпорядок на основі внесених користувачем даних.

Постановка задачі

Люди які розплановують свій час часто користуються різними методами та способами цього планування, вони використовують різноманітні програми та застосування, починаючи з папірців на холодильнику і завершуючи різноманітними текстовими редакторами та різними щоденниками. По суті такі програми являються так званими програмами-органайзерами, тобто це програми що допомогають підвищити ефективність будь-якої праці, як в особистому житті так і професійної. Але суть процесу планування завжди одна, людина сама розміщує свої задачі в певному порядку, інколи розкидуючи їх по різним дням. Зазвичай, для такого планування витрачається багато часу, адже кожну задачу потрібно вносити наперед, вручну, що не є ефективним.

Так для планування свого робочого часу, скажімо, на 2 тижні потрібно витратити близько години. Також при такому плануванні рідко враховуються такі фактори як: час на перерви, протяжність цих перерв, час на те щоб «переналаштувати» свій розум на нову задачу, адже це може викликати потреби «вникати» у проблему заново. Саме тому набагато ефективніше розпланувати час так, щоб виконання 1 задачі завершувалося або на перерву, або по завершенню робочого часу.

Також слід враховувати усі «дедлайни» для кожної з задач, що в свою чергу теж потребує час на осмислення.

При плануванні гнучких графіків виникають ситуації коли потрібно максимально ефективно розпланувати свій час, враховуючи усі можливі ситуації що можуть завадити виконанню роботи. Набагато краще коли в щоденнику стоїть не просто потрібно зробити ще 5 проектів за 15 днів, а розписано що конкретно потрібно робити щодня, для виключення ситуації коли потрібно зробити всі ті самі 5 проектів, але вже за 2 дні. Адже є багато досліджень на основі яких можна складати зручні розклади робочого часу.

Отже метою статті є клієнт-серверна програма яка призначена для автоматизації управляння робочим часом.

Присвоєння задачам пріоритетів по матриці Ейзенхауера

Оскільки немає сенсу змушуватии користувача власноруч встановлювати наскільки високий пріоритет має кожна з задач, тож будемо вважати що він у всіх задач високий, але користувач матиме змогу його знизити.

При введенні інформації про кінцеві терміни задачі користувач також буде вносити інформацію про те що ця задача є надважливою, або не є важливою, якщо такої інформації внесено не було, то вона вважається важливою.

В матриці пріоритетів Ейзенхауера є 4 сектори («важливі й термінові», «неважливі й термінові», «важливі й нетермінові», «неважливі й не термінові»), але використовуються в нашій программі з них лише 3 з них: «важливі й термінові», «неважливі й термінові», «важливі й нетермінові». Оскільки сегмент «неважливі й не термінові», не доцільно вносити до розкладу робочого часу, так як робочих задач він в собі по означенню не несе. Він заповнений різними діями накшталт поговорити з другом по телефону, пограти в ігри тощо.., і не несе в собі ніякого ефективного навантаження [3].

Отже пріоритет задачі та її положення в матриці розраховується за формулою:

$$P = \alpha + \beta$$

де: P – пріоритет задачі, α – термін виконання задачі, β – важливість виконання задачі.

Важливість визначається на основі інформації внесеної користувачем, і має 3 рівні. По замовчуванню всі задачі важливі, але користувач може додатково позначити задачу як надважливу, або не важливу (якщо це наприклад його хобі).

Терміновість виконання залежить від часу до «дедлайну» задачі, що ближче термін здачі, то вища терміновість, відповідно, що далі термін здачі – то терміновість нижча.

Терміновість виконання розраховується відповідно до вище сказаного по формулі:

$$\alpha = \frac{t}{day}$$

де: day – це кількість робочих днів до завершення терміну виконання роботи,

t – це час на виконання роботи (теж зводиться до робочих днів).

Саме на основі цих розрахунків визначається пріоритет задач при складанні розкладу, в першу чергу йдуть задачі що належать до групи «важливі й термінові», до цієї групи входять не більше 2-3 задач, і вони стоять на першому місці при складанні розпорядку. Далі йдуть задачі виконання яких термінове, але не обов'язкове. І останніми в розкладі йдуть задачі що на данний момент не є терміновими але являються важливими.

Розклад складається для кожного дня окремо, тож терміновість задач зростає, і по мірі завершення одних завдань до групи «важливі й термінові» будуть потрапляти інші.

Аналіз досліджень

Актором системи є користувач. Робота з програмою починається з внесення певного переліку своїх справ на період часу. Якщо не було внесено нових справ, і раніше програму вже використовували виводиться згенерований раніше розклад. Якщо ж користувач використовує програму вперше то клієнтська частина відправить інформацію клієнта на серверну програму, що опрацює внесені дані та на їх основі згенерує кілька варіантів розкладу. Далі чорнові варіанти розкладу надходять до клієнта, і він має змогу внести певні корективи, або зберегти варіант який йому більше сподобається. Наведемо діаграму діяльності процесу створення розкладу робочого часу (рис. 1) та діаграму послідовності (рис. 2) процесу створення розкладу робочого часу. На діаграмі послідовності зображені основні об'єкти і актори, що взаємодіють між собою на певному проміжку часу:

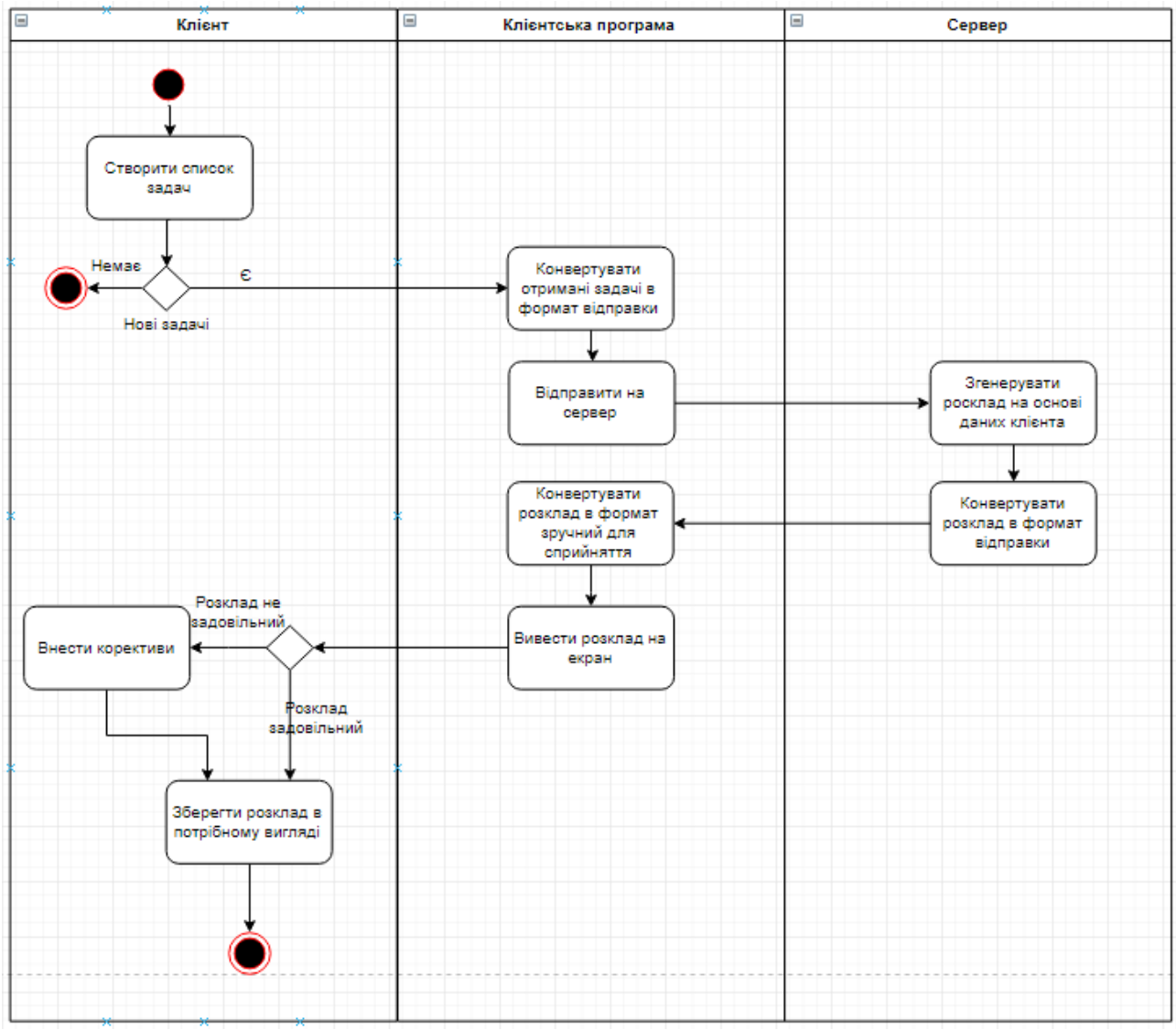


Рис.1. ─ Діаграма діяльності процесу процесу створення розкладу робочого часу

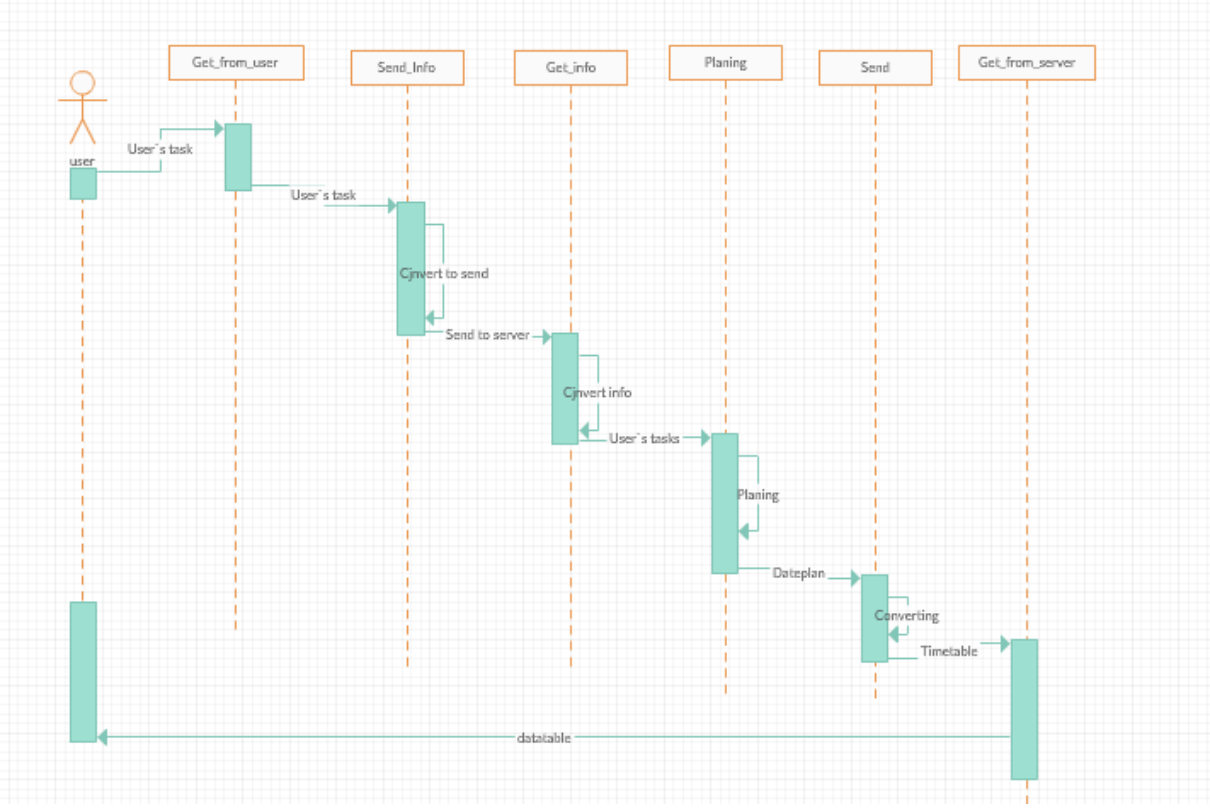


Рис.2. ─ Діаграма послідовності процесу створення розкладу робочого часу

Вхідні та вихідні дані

Вхідні дані для аналітичної системи з оптимізації та управлінням використання робочого часу надходять від користувача системи у вигляді наступних даних:

Налаштування складається з задач описаних наступним чином:

- Словесний опис задачі;
- Орієнтовний час її виконання;
- Остаточний термін виконання задачі;
- Коефіцієнт відхилення дороги;
- Пріоритетність задачі;
- День початку виконання задачі;

Загальні налаштування системи генерації розпорядків:

- Вихідні дні;
- Періоди які вважаються робочим часом;
- Кількість перерв, та їх довжина.

На виході (після завершення генерації) створюється кілька варіантів попереднього розкладу, на основі яких користувач зможе швидко створити ідеальний для себе розпорядок.

Висновки

В даній статті було обґрунтовано доцільність розробки клієнт-серверної аналітичної системи з оптимізації та управлінням використання робочого часу. Було визначено структуру системи, показано, що алгоритмічне повинне включати в себе використання матриці пріоритетів Ейзенхауера. Було показано структуру системи, та основні її модулі.

Список літератури

1. Jetbrains. Products [Електронний ресурс] / JetBrains. – 2019. – Режим доступу до ресурсу: https://jetbrains.ru/products/youtrack/.

2. LeaderTask.[Електронний ресурс] / Organizer LeaderTask – 2019. – Режим доступу до ресурсу: https://www.leadertask.com/.

3. Матриця Ейзенхауера для складання списку справ. [Електронний ресурс] – Режим доступу до ресурсу: http://how-to-do.org/matrytsa-ejzenhauera/.

УДК 656.222

ХРАМЧЕНКО М.С.
МУХА І.П.

МЕТОД ПОБУДОВИ НАВІГАЦІЙНИХ МАРШРУТІВ З УРАХУВАННЯМ ЗАДАНОЇ МНОЖИНИ КРИТЕРІЇВ

Розглянута задача побудови автомобільних навігаційних маршрутів у багатокритеріальній постановці. Запропоновано метод побудови навігаційних маршрутів з урахуванням декількох критеріїв, який грунтується на використанні метода згортанні критеріїв і удосконаленого алгоритму А*.

The problem of constructing automobile navigation routes in a multi-criteria setting is considered. The method of construction of navigation routes based on several criteria is proposed, which is based on the use of the method of curtailment of the criteria and the improved algorithm A *.

Постановка проблеми.

Рівень розвитку сучасних технологій дозволяє обладнувати мобільні пристрої (мобільні телефони, кишенькові та планшетні комп'ютери) засобами доступу до бездротового мобільного Інтернету та до глобальної системи супутникової навігації GPS (англ. Global Positioning System). Це дає змогу організуватиназемну навігацію рухомих об'єктів, зокрема, автомобільну, використовуючи мобільні навігаційні додатки.

Однією із найбільш важливих задач таких навігаторів є побудова оптимального маршруту руху об'єкта до заданої точки.

Наявні способи реалізації даного функціоналу в існуючих програмних засобах (таких як GoogleMaps[1], YandexMaps[2], AppleMaps[3] та інші) передбачають при побудові маршруту можливість обрати на власний розсуд, як критерій оптимальності, найкоротшу відстань або найменший час. За замовчуванням система обирає найшвидший маршрут. Якщо існує два маршрути з однаковим часом проходження, система автоматично обирає найкоротший із них та пропонує альтернативний варіант.

Значним недоліком таких мобільних навігаторів є обмеженість вибору критеріїв оптимізації при побудові маршруту. Наприклад, відсутня можливість вибору маршруту з урахуванням якості дорожнього покриття, завантаженості доріг, якостіїх освітленості, інформації про аварійні ситуації, ремонтні роботи, пробки, закриття дороги або зміну напрямку руху, допустимого відхилення від найменшого часу або найкоротшої дистанції тощо.

Тому актуальною є задача розробки метода побудови навігаційних маршрутів з можливістю урахування декількох критеріїв оптимальності, визначених користувачем.

Огляд існуючих методів.

Традиційним підходом до побудови навігаційних маршрутів є моделювання цифрової географічної карти (мережі доріг) у вигляді зваженого графа, вершини якого представляють собою місця можливих змін напрямку доріг (дорожні розв'язки, перехрестя, повороти дороги тощо), а ребра – самі дороги, і вирішення задачі пошуку найкоротшого шляху між заданими вершинами даної мережі.

Орієнтовані ребра у такому графі відповідають вулицям з одностороннім рухом, неорієнтовані - вулицям з двостороннім рухом. Ваги ребер можуть визначати відстані між окремими вершинами, час проїзду по ділянці дороги тощо. Відповідно, основними критеріями пошуку у такій мережевій моделі є, як правило, найкоротший шлях або найшвидший маршрут.

Необхідність урахування декількох критеріїв оптимальності одночасно зводить дану задачу до задачі багатокритеріальної оптимізації [4].

Формулюється дана задача у такий спосіб:

$$\min_{\overline{x}}\{f_1(\overline{x}), f_2(\overline{x}), \dots, f_k(\overline{x})\},\ \overline{x} \in S \quad (1)$$

де $f_1(\overline{x}), f_2(\overline{x}), \dots, f_k(\overline{x})$ - скалярні функції векторного аргументу $\overline{x} = (x_1, x_2 \dots, x_n)^T$, кожна з яких є математичним виразом одного критерію оптимальності, S – множина допустимих рішень.

Вирішення даної задачі полягає в пошуку вектора цільових змінних $\overline{x}$, що задовольняє накладеним обмеженням, і оптимізує векторну функцію, елементи якої відповідають цільовим функціям $f_1(\overline{x}),\ f_2(\overline{x}), \dots,\ f_k(\overline{x})$.

Так як в моделі (1) використовується векторна цільова функція, таку задачу ще називають задачею векторної оптимізації [4]. Основною особливістю даної задачі, як часткового випадку багатокритеріальної оптимізації, є наявність невизначеності, яка полягає у тому, що невідомо, яким критеріям треба віддати перевагу і в якій мірі [4].

Деякий розв'язок $\overline{x}$* задачі (1) назива- ють ефективним рішенням даної задачі, якщо таке рішення не можна поліпшити по якомусь із критеріїв, не погіршивши при цьому значення інших критеріїв [5]. Множину ефективних рішень називають множиною Парето. Оптимальне рішення слід шукати лише серед елементів множини Парето [4]. Це дозволяє звузити клас можливих претендентів на остаточне рішення і виключити з розгляду завідомо неконкурентоспроможні варіанти [4].

Тому термін «оптимізувати» в задачі багатокритеріальної оптимізації означає: знайти таке рішення, при якому значення цільових функцій $f_1(\overline{x}),\ f_2(\overline{x}), \dots,\ f_k(\overline{x})$ були б прийнятними для постановника задачі.

Існуючі підходи до вирішення задачі багатокритеріальної оптимізації передбачають використання:

– методів, заснованих на згортанні критеріїв;

– методів, які використовують обмеження на критерії;

– методів цільового програмування;

– методів, заснованих на знаходженні компромісного рішення [4].

Замість вихідної багатокритеріальної задачі відповідно до обраного методу, формується задача, що її заміщає. До складу такої задачі входить один критерій, а до вихідної системи обмежень додається одне або декілька додаткових обмежень. Розв'язок такої задачі називається субоптимальним [4].

Метод побудови навігаційних маршрутів.

Задача побудови навігаційних маршрутів з можливістю урахування декількох критеріїв оптимальності, як задача мережевого аналізу, передбачає моделювання системи (у нашому випадку - цифрової географічної карти) у вигляді графа та подальший аналіз даного графа з метою пошуку найкоротшого (найоптимальнішого за вказаними критеріями) маршруту між двома його вершинами.

Оскільки дана задача передбачає можливість урахування декількох критеріїв оптимальності, то, щоб враховувати цю інформацію при аналізі мережі, треба розглядати зважені графи.

Зваженим є граф, кожному ребру якого ставиться у відповідність число, яке називається вагою цього ребра.

Математично постановка даної задачі може бути сформульована наступним чином:

Нехай заданий зв’язний зважений граф $G = (V, R)$, вершини V якого відповідають вузлам цифрової географічної карти (місцям можливих змін напрямку доріг - перехрестям, поворотам дороги тощо), ребра R – шляхам, що з’єднують ці вузли, а ваги ребер w – протяжностям цих шляхів.

Також кожне ребро $r = (u, v) \in R$, $u \in V, v \in V$ має свій коефіцієнт якості $k(u, v) \in K, k = (0, 1]$, де K – це множина коефіцієнтів якості ребер графа G.

Маршрутом у графі є послідовність ребер $r \in R$, які утворюють неперервну послідовність вузлів:

$$\forall r_i \in R: end(r_{i-1}) = start(r_i),$$
$$end(r_i) = start(r_{i+1}),$$

де $start(r) \in U$ - це номер вузла, з якого виходить ребро r, $end(r) \in U$ - це номер вузла, в яке виходить ребро r.

Необхідно визначити маршрут мінімальної довжини між двома заданими вершинами, а також усі альтернативні маршрути між цими ж вершинами, відхилення довжин яких від мінімальної знаходиться в заданому діапазоні.

Для того, щоб побудувати навігаційний маршрут з урахуванням декількох критеріїв оптимальності, пропонується за допомогою методів багатокритеріальної оптимізації перезважити ребра графу з урахуванням вказаних критеріїв.

Для вирішення даної задачі можна використати метод згортання критеріїв.

Відповідно до даного методу, багатокритеріальна задача зводиться до задачі з одним зваженим критерієм (2):

$$\min_{x\in X} \sum_{k=1}^{p} \lambda_k f_k(x) \qquad (2),$$
$$\sum_{k=1}^{p} \lambda_k = 1,$$
$$\lambda_k \geq 0,$$

де p – кількість критеріїв, f – функція відповідного критерію, λ - ваговий коефіцієнт відповідного критерію, X – множина допустимих рішень при різних значеннях λ_k.

Результуючий ваговий коефіцієнт якості кожного ребра графа при цьому підраховується за формулою (3):

$$k(r) = \sum_{k=1}^{p} \lambda_k f_k(r) \qquad (3).$$

З урахуванням вищезазначеного, результуючою вагою ребра *Wi* між двома вершинами графа є добуток отриманого вагового коефіцієнту якості ребра $r_i \in R$, що визначаються за заданими критеріями, на довжину даного ребра *w* (4):

$$Wr_i = w(r_i)k(r_i), r_i \in R, \qquad (4)$$

де *i* – індекс ребра.

Результуючою вагою маршруту W_j між двома довільними вершинами графа є сума ваг кожного ребра Wr_i даного шляху:

$$W(R_j) = \sum_i W(r_i). \qquad (5)$$

де j – індекс маршруту, R_j – це множина ребер r_i, що утворюють маршрут W_j.

Згідно метода згортання критеріїв, оптимальним маршрутом до будь-якої вершини графу є шлях з мінімальною вагою (6):

$$W(R_o) = \sum_i w(r_i)k(r_i) \to min,$$
$$\forall R_j \in T,\ W(R_o) = \min_{R_j\in T} W(R_j) =$$
$$\min_{R_j\in T} \sum_i w(r_i) \sum_{k=1}^{p} \lambda_{ik} f_{ik}(r_i),$$

де R_o – це множина ребер r_i, що утворюють маршрут мінімальної довжини, T – множина допустимих маршрутів.

Алгоритм знаходження найкоротшого шляху.

Для визначення маршруту мінімальної довжини пропонується застосовувати удосконалений варіант відомого алгоритму A*, що належить до евристичних алгоритмів пошуку.

Алгоритм A* заснований на розстановці позначок на вершинах графа, які представляють собою скалярні величини, що дорівнюють сумі вагових коефіцієнтів ребер графа на шляху від початкової точки до поточної.

Модифікація даного алгоритму полягає у тому, що на кожній ітерації процесу пошуку необхідно не намагатися вибрати значення позначки з меншим значенням, що відповідає поточній вершині графа, а створювати нову позначку, доповнюючи нею множину позначок поточної вершини. Вихідними даними модифікованого алгоритму буде множина маршрутів, яку можна назвати множиною субоптимальних маршрутів.

Так як алгорим працює з «переваженим графом» в залежності від критеріїв оптимальності, необхідно перерарухавати значення оригінальної довжини кожного маршруту та порівняти їх з найкоротшим. Оригінальна довжина субоптимальних маршрутів не повинна перевищувати максимально допустиму похибку - ε, що є вхідним параметром алгоритму.

Алгоритм пошуку субоптимальних маршрутів знаходить альтернативні шляхи, довжини яких перевищують довжину найкоротшого маршруту в межах допустимого відхилення ε:

$$\forall R_S \in T, \frac{W(R_o)}{W(R_j)} < \varepsilon \qquad (7)$$

де $R_S -$ субоптимальний маршрут.

Алгоритм знаходження найкоротшого шляху.

Представимо алгоритм пошуку субоптимальних шляхів на прикладі графу, показаного на рис 1, де початкова точка – вершина 1, кінцева точка – вершина 4, допустима похибка – 20%.

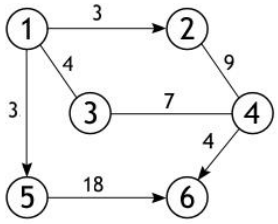


Рис. 1

Введемо 2 критерія якості для ребер даного графа. Нехай перший критерій має пріоритет 0.7, а другий відповідно 0.3.

Задамо значення критеріїв для кожного ребра графа.

Критерій 1:

1-2	2-4	1-3	3-4	1-5	5-6	4-6
0.25	1	1	1	0.7	0.23	0.32

Критерій 2:

1-2	2-4	1-3	3-4	1-5	5-6	4-6
0.53	1	1	1	0.6	0.94	0.92

Крок 1. Підрахуємо вагові коефіцієнти для кожного ребра за формулою (3) та ваги ребер з урахуванням цих критеріїв за формулою (4) (рис. 2).

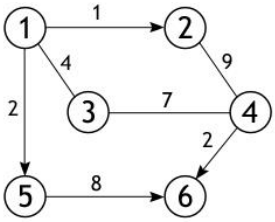


Рис. 2

Крок 2. Початковій вершині 1 встановимо позначку із значенням 0.

Крок 3. У вершини 1 рівно три сусіда: вершини 2, 3, 5. Щоб обчислити довжину шляху до кожного з них потрібно скласти ваги ребер, що лежать між вершинами 1 і 2, 1 і 3, 1 і 5 зі значенням позначки першої вершини (тобто, з 0) (рис. 3):

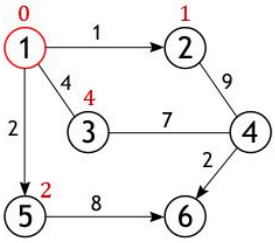


Рис. 3

Крок 4. Поточна (активна) вершина позначається як відвідана, статус «активної» переходить до однієї із її сусідок, а саме до вершини 2, оскільки вона найближча по довжині ребра до раніше активної вершини (рис. 4).

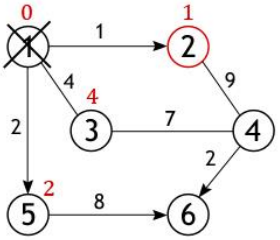


Рис. 4

Крок 5. У вершини 2 всього один не розглянутий сусід, відстань до якого з неї дорівнює 9. Необхідно обчислити довжину шляху з початкової вершини 1, для чого потрібно скласти величину позначки вершини 2 з вагою ребра до вершини 4. До позначок вершини 4 додається позначка зі значенням 10 (рис. 5).

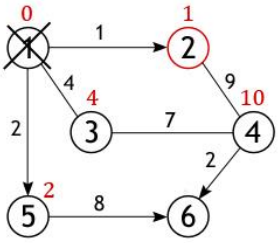


Рис. 5

Крок 6. Вершина 2 перестає бути активною, і видаляється зі списку не відвіданих. Тепер тим же способом досліджуються сусіди вершини 5, і обчислюється відстань до них (рис. 6). У вершини 6 немає не розглянутих сусідів.

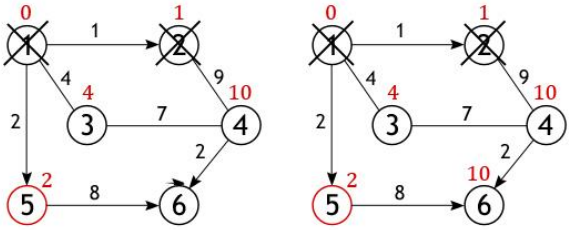


Рис. 6

Крок 7. Аналогічно розглядаємо вершину 3 та її сусідів. При проході через вершину 3 до кінцевої точки, вершина 4 отримує додаткову позначку – 11 (рис. 7).

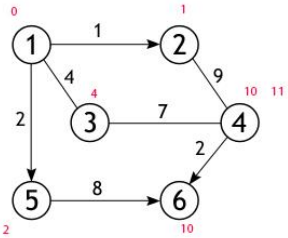


Рис. 7

Крок 8. З урахуванням вказаних критеріїв маємо субоптимальні маршрути: 1-2-4 (1 маршрут) з довжиною 10 та 1-3-4 (2 маршрут) з довжиною 11.

Крок 9. Підрахуємо оригінальні довжини даних шляхів:

1-2-4 = 12

1-3-4 = 11

Крок 10. Так як оригінальна довжина субоптимального маршруту 1 більша за довжину маршруту 2, але в межах допустимої похибки (20%), то вважаємо, що маршрут 1 є оптимальним по заданим критеріям.

Висновки.

Для вирішення задачі побудови навігаційних маршрутів з можливістю урахування декількох критеріїв оптимальності доцільним є використання методів, заснованих на згортанні критеріїв.

Запропоновано метод побудови навігаційних маршрутів з урахуванням декількох критеріїв оптимальності, що базується на використанні удосконаленого алгоритму A*.

Використання такого алгоритму дозволяє визначити декілька субоптимальних рішень, для яких необхідно підрахувати оригінальні довжини маршруту та відкинуті ті, що перевищують максимально допустиме відхилення.

Список літератури

1. GoogleMaps. [Електронний ресурс] – Режим доступу: https://www.google.ru/maps
2. YandexMaps. [Електронний ресурс] – Режим доступу: https://yandex.ru/maps/
3. AppleMaps. [Електронний ресурс] – Режим доступу: https://mapsconnect.apple.com/
4. Ю. В. Чибісов, Ю. С. Шульга. Застосування методів багатокритеріальної оптимізації для вирішення задачі розподілу вагонів по вантажним фронтам. //«Транспортні системи та технології перевезень». Збірник наукових праць ДНУЗТ ім. акад. В. Лазаряна. -Вип. 7. - 2014. - С. 65-72.
5. Постановка задачи векторной оптимизации [Електронний ресурс]. – Режим доступу: http://edu.nstu.ru/courses/mo_tpr/files/5.html
6. The A* Algorithm. [Електронний ресурс] – Режим доступу: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html#algorithms
7. Клименко В. Г. Багатокритеріальна оптимізація на графах [Текст]: наук. вид.: [монографія] / В. Г. Клименко. - 7-ме вид., допов. - Харків: Федорко, 2018. - 727 с.

УДК 004

СБОРИК А.Ю.,
ТЄЛИШЕВА Т. О.

Депозитний модуль інформаційної системи АБС “БАРС” для підтримки та розвитку малого та середнього бізнесу

Стаття присвячена розробленню депозитного модуля в складі банківської інформаційної системи для підтримки малого та середнього бізнесу для полегшення процесу оформлення депозитів та забезпечення більшої надійності шляхом заміни рукописних документів електронними, що формуються на основі введених даних. Надані аналіз бізнес-процесів оформлення депозитів, структури вихідних документів та структура бази даних для збереження даних про депозити.

ДЕПОЗИТ, БАЗА ДАНИХ, БІЗНЕС-ПРОЦЕС, ПРОГРАМНИЙ МОДУЛЬ, ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ, МАЛИЙ БІЗНЕС, СЕРЕДНІЙ БІЗНЕС, ЗВ’ЯЗКИ МІЖ ТАБЛИЦЯМИ БД

The subject of the article is development of a deposit module as a way of supporting small and medium-scale businesses, by facilitating the registration of deposits and ensuring greater reliability of deposit registration by replacing handwritten documents with electronic ones, which are formed on the basis of the data entered. The article describes the business process of deposit registration, the structure of source documents and the structure of the database to store the data on deposits.

DEPOSIT, DATABASE, BUSINESS-PROCESS, SOFTWARE MODULE, USE-CASE DIAGRAMM, SMALL BUSINESS, MEDIUM-SCALE BUSINESS, RELATIONSHIPS BETWEEN TABLES IN DB

Вступ

Сьогодні в економіці розвинених країн велику роль відіграють малий та середній бізнес. Зокрема, їх частка у ВВП багатьох європейських країн перевищує 50 % (Італія – 70 %, Німеччина – 60 %, Франція – 52 %), в Польщі – 47 % ВВП. В Україні ж мале підприємництво знаходиться лише на етапі становлення (частка внеску в ВВП становить 12-14 %) та потребує значної підтримки[2].

Депозитні внески[1], як спосіб прибутку та зберігання грошей для малого та середнього бізнесу є гарним капіталовкладенням.

Для підтримки та розвитку малого та середнього бізнесу банківська інформаційна система (ІС) АБС розширюється за рахунок створення модуля для полегшення оформлення та забезпечення більшої надійності оформлення депозитів шляхом заміни рукописних документів електронними, що формуються на основі введених даних.

Метою реалізації депозитного модуля є збільшення швидкості та надійності роботи відділу банку з оформлення депозитів за рахунок автоматизації процесу оформлення депозитів та зберігання даних про клієнтів.

При автоматизації даного процесу будуть досягнуті наступні підцілі:

- зменшення часу на відкриття депозиту;
- підвищення комфортності користування системою для працівників банку;
- підвищення надійності при оформленні депозиту працівником і клієнтом банка.

Призначенням депозитного модуля, окрім підтримки та розвитку малого та середнього бізнесу, є аналіз популярності наявних депозитних продуктів: розрахунок прогнозованої кінцевої суми, отриманої від депозитного вкладу та дати повернення депозиту, побудова діаграм і т.і.

Для досягнення поставлених цілей мають бути вирішені такі задачі:

- формування анкети клієнта, заяви на відкриття депозиту, договору на основі введених даних про клієнта та введення необхідної інформації;
- пошук клієнта в базі даних банку;
- запровадження вибору типу депозиту зі списку наявних депозитних продуктів, введення даних про депозит;
- розрахунок прибутку отриманого від депозитного вкладу, знаходження дати повернення депозиту;
- підрахунок частості вибору певного депозитного продукту.

В даний час існує декілька модулів в автоматизованих банківських системах Грант[3], ИСАОД[4] та ін. Основною проблемою використання існуючих рішень є відсутність перегляду статистики по існуючим депозитним продуктам, а також неповнота функціоналу та застаріли технології, що використовуються в деяких з них.

Бізнес-процес оформлення депозиту наведено на рисунку 1 за допомогою діаграми IDEF0.

Рисунок 1. Діаграма бізнес-процесу оформлення депозиту

При відкритті нового депозиту оператор здійснює пошук клієнта в базі даних банку. В разі наявності клієнта в базі даних дані автоматично переносяться до форми введення, а в разі відсутності оператор вносить дані до форми введення. Також оператор вводить дані про депозит та обирає тип депозиту. Дані для представлення можливих типів депозитів беруться з довідників депозитів, що знаходяться в базі даних банку Після введення всіх даних оператором фронт офісу дані про депозит зберігаються до бази даних банку та відправляються оператору бек офісу для підтвердження, також оператор фронт офісу має можливість надрукувати заяву на відкриття депозиту та за необхідності анкету клієнта. Після підтвердження даних депозиту оператором бек офісу, оператор фронт офісу може надрукувати договір з даними, що беруться з бази даних банку, що дає можливість клієнту банку не витрачати час на заповнення договору власноруч, а лише поставити підпис.

Акторами у системі є оператор фронт офісу та оператор бек офісу.

Оператор фронт офісу має можливість перегляду відкритих за сьогодні договорів, всіх підтверджених договорів, та відкриває нові депозити та має можливість друку анкети клієнта, заяви на відкриття депозиту, депозитного договору

Оператор бек офісу підтверджує дані про оформлення депозиту, введені оператором фронт офісу.

Діаграма варіантів використання наведена нарисунку 1.2.

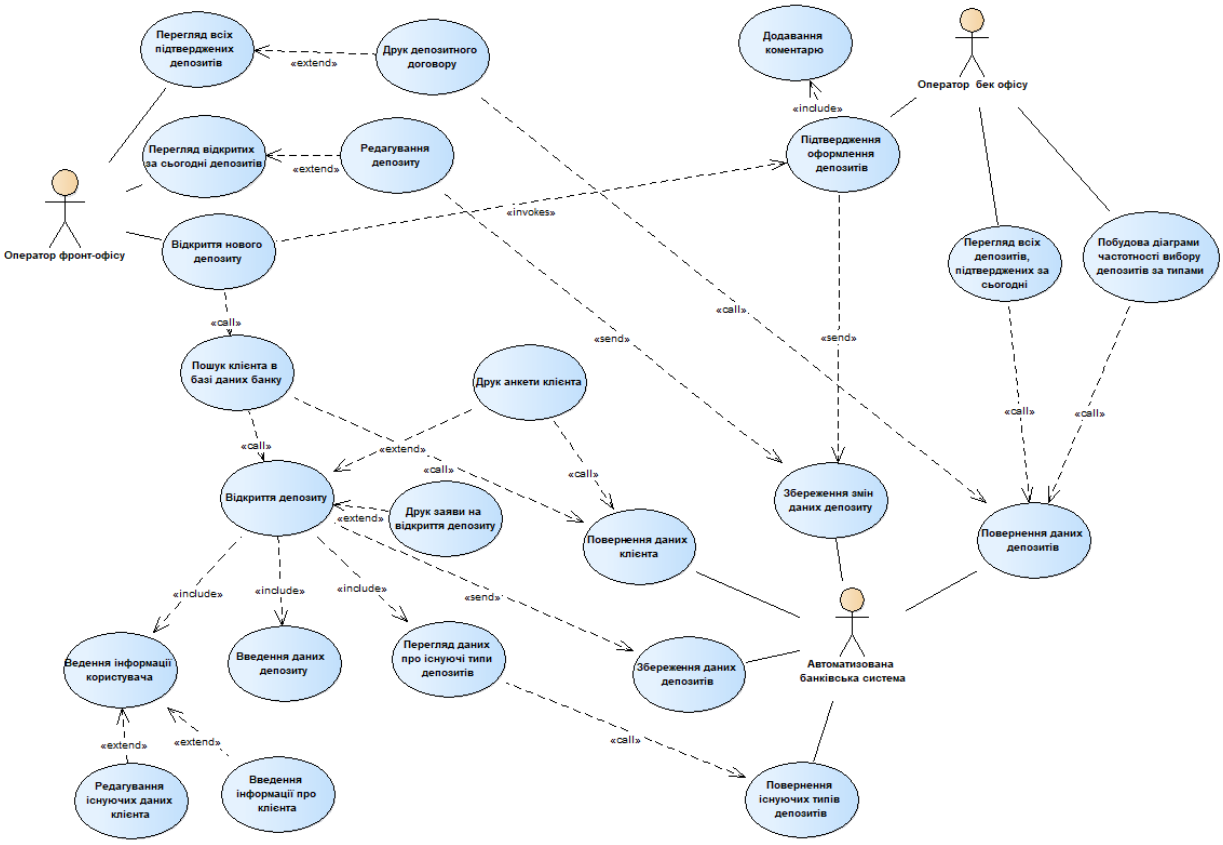


Рисунок 2. Діаграма варіантів використання

Розглянемо структуру даних, яка потрібна для відкриття депозиту, а саме дані про клієнта, дані про тип депозитного продукту, дані про депозит.

Дані про клієнта:

- найменування клієнта банку – назва юридичної особи;
- в особі – прізвище ім'я по-батькові представника юридичної особи;
- код ідентифікації – код ІНН або код за ЕРДПОУ;
- місце реєстрації – місце реєстрації представника юридичної особи;
- номер телефону – номер телефону представника юридичної особи;

–номер банківського рахунку – номер банківського рахунку юридичної особи для списання коштів та повернення коштів після закінчення депозитного строку;
–номер банківського рахунку для депозита – номер банківського рахунку для виплати відсотків по депозиту.

Дані про тип депозитного продукту:

–назва – назва типу депозитного продукту;
–автопролонгація – наявність можливості автопролонгації для даного депозитного продукту;
– можливість поповнення – наявність можливості поповнення для даного депозитного продукту;
– можливість дострокового розірвання – наявність можливості дострокового розірвання для даного депозитного продукту;
–відсоткова ставка – відсоткова ставка для даного депозитного продукту.

Дані про депозит:

–дата початку – дата початку депозитного строку;
–дата закінчення – дата закінчення депозитного строку;
–сума вкладу – сума вкладу при відкритті депозиту;
–валюта – валюта в якій відкривається депозит;
–клієнт – клієнт, що відкриває депозит;
–тип – тип обраного депозитного продукту.

Вихідними документами є:

– анкета клієнта;
– заява на відкриття договору;
– депозитний договір.

У таблиці 1 наведений опис вихідних документів.

Таблиця 1 – Опис вихідних документів

№	Найменування	Кодове позначення	Реквізити
1	Анкета клієнта	АК	- найменування клієнта банку; - в особі; - код ідентифікації; - місце реєстрації; - номер телефону; - номер банківського рахунку; - номер банківського рахунку для депозиту; - дата; - підпис клієнта.
2	Заява на відкриття договору	ЗВД	- найменування клієнта банку; - в особі; - код ідентифікації; - місце реєстрації; - номер телефону; - номер; - банківського рахунку; - номер банківського рахунку для депозиту; - назва типу депозиту; - автопролонгація; - можливість поповнення;

№	Найменування	Кодове позначення	Реквізити
			- можливість дострокового розірвання; - відсоткова ставка; - дата початку; - дата закінчення; - сума вкладу; - валюта; - дата; - підпис клієнта.
3	Депозитний договір	ДД	- найменування клієнта банку; - в особі; - код ідентифікації; - місце реєстрації; - номер телефону; - номер банківського рахунку; - номер банківського рахунку для депозита; - умови договору; - назва типу депозиту; - автопролонгація; - можливість поповнення; - можливість дострокового розірвання; - відсоткова ставка; - дата початку; - дата закінчення; - сума вкладу; - валюта; - дата; - підпис працівника банку; - підпис клієнта.

Опис структури бази даних

Таблиця DepositInfoes (рис. 2) містить інформацію про можливі типи депозитних продуктів, таблиця Deposites – інформацію про відкриті депозити, ClientInfoes – містить інформацію про клієнтів. Таблиці DepositInfoes та Deposites мають зв’язок один до одного або нуля, тобто Deposites обов’язково містить тип депозиту, а таблиця DepositInfoes може не містити посилання на Deposites. Аналогічний зв’язок мають таблиці Deposites та DepositInfoes.

Було проведено випробування депозитного модуля з метою перевірки відповідності функцій модуля відкриття депозитів для малого та середнього бізнесу вимогам технічного завдання.

Тестування проводилась за допомогою розробки тестових сценаріїв та порівняння очікуваних результатів з отриманими.

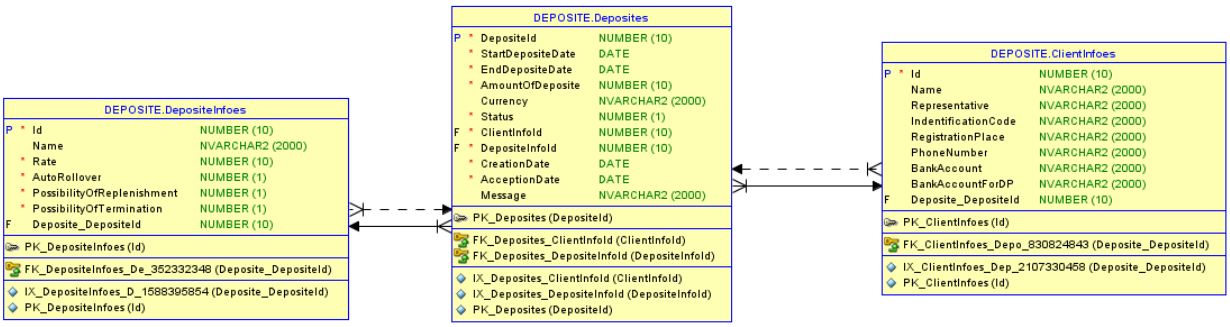


Рисунок 2– Схема БД

Висновок

В даній статті було описано розроблений процес автоматизованого оформлення депозиту, наведені діаграми в нотаціях IDEF0 та UML, визначено структуру документів, що створюються на основі введених даних, описано структуру бази даних та зв'язки між таблицями. Оскільки час оформлення депозиту зменшується, а надійність збільшується, то це сприяє підтримці малого та середнього бізнесу за рахунок більшої пропускної здатності відділень, а тим самим більше підприємців можуть зроби вклад з метою збереження коштів, або прибутку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Банківська енциклопедія / С.Г. Арбузов, Ю.В. Колобов, В.І. Міщенко, С.В. Науменкова. – К. : Центр наукових досліджень Національного банку України : Знання, 2011. – 504 с. – (Інституційні засади розвитку банківської системи України)
2. ТЕНДЕНЦІЇ РОЗВИТКУ МАЛОГО БІЗНЕСУ В УКРАЇНІ – [Електроний ресурс] // Режим доступу: http://conf.management.fmm.kpi.ua/proc/article/view/62601
3. Автоматизированная банковская система – [Електроний ресурс] // Режим доступу: http://www.banksoft.com.ua/project/bis-grant/
Фронт-офисные системы – [Електроний ресурс] // Режим доступу: http://www.ibis.ua/products/front_office/