НАЦІОНАЛЬНА
АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 1 ***січень-березень*** ***2024***

Заснований у березні 1999 р.

ЗМІСТ

Свідоцтво про державну реєстрацію КВ № 7490 від 01.07.2003

Науковий журнал “Проблеми програмування” занесений до переліку наукових фахових видань України, в яких можуть публікуватися основні результати дисертаційних робіт.

ISSN 1727-4907

NATIONAL ACADEMY
OF SCIENCES OF UKRAINE
INSTITUTE OF SOFTWARE SYSTEMS

PROBLEMS OF PROGRAMMING

scientific journal

№ 1 ***January – March*** ***2024***

Founded in March, 1999

CONTENT

УДК 004.424 http://doi.org/10.15407/pp2024.01.003

Р.С. Шевченко, А.Ю. Дорошенко, О.А. Яценко

ВБУДУВАННЯ СІМЕЙСТВА ЛОГІЧНИХ МОВ ІЗ МОЖЛИВОСТЯМИ ПЕРЕПРОГРАМУВАННЯ МОНАДИЧНОЇ УНІФІКАЦІЇ В SCALA

У статті запропонована структура для вбудовування методів логічного програмування та програмування в обмеженнях у мову Scala шляхом побудови логічної предметно-орієнтованої мови навколо уніфікації типізованої логіки на основі монад. Відмінності в можливостях логічних механізмів можна виразити як підкласи монади уніфікації. Такий спосіб дає змогу генерувати одну реалізацію налаштовуваної уніфікації для вбудовування різних логічних систем у Scala та використовувати вбудовані сторонні проблемно-орієнтовані мови у логічних виразах. Монадичний прикладний програмний інтерфейс надає розробнику програми простий та інтуїтивно зрозумілий інструмент для реалізації власної логіки всередині уніфікації.

Ключові слова: вбудовування, декларативне програмування, логічне програмування, монада, предметно-орієнтована мова, терм, уніфікація, Scala.

Вступ

Декларативне програмування є важливим елементом інженерії програмування. Сьогодні більшість декларативних мов є переважно галузевими техніками, причому в більшості областей логічне програмування загального призначення, як правило, не застосовується. Деякі задачі, такі як взаємодія з користувачами або інтеграція з іншими програмами, завжди будуть поза формалізацією, тому що суть такої роботи — побічний ефект, який краще описати операційно. Проте найпоширенішим підходом є гібридний, коли розробники використовують високорівневі методи для однієї частини системи та операційні — для інших. Для цього потрібна підтримка базової інфраструктури, зокрема:

– незалежні від мови моделі сутностей;

– вбудовування декларативних методів у недекларативні мови.

Перший випадок використовується з архітектурою мікросервісів, коли ми можемо розділити велику систему на грубозернисті підсистеми, де кожна підсистема має бажаний стиль. Однак поділ системи на мікросервіси, як правило, здійснюється не за технологіями, а за бізнес-ознаками, тому використання архітектури на основі мікросервісів часто є більш організаційним, аніж технічним рішенням.

Другий випадок — вбудовування елементів декларативного програмування в операційну семантику. Тут ми можемо побачити в ландшафті нові мультипарадигмальні мови програмування, такі як Flix [1, 2], що поєднують логічне, функціональне та імперативне програмування, або ж Logtalk [3], який додає об’єктну орієнтацію до мови Prolog та більшу тенденцію до вбудовування декларативних методів програмування в існуючі мови. Декларативна програма представлена тут як об’єкт у системі об’єктів основною мовою, зазвичай створений у певного виду вбудованій предметно-орієнтованій мові (Embedded Domain-Specific Language, EDSL). Сьогодні це поширена методика. TermWare [4, 5] вбудовує програмування на основі правил у Java, 2P-Kt [6] вбудовує Prolog у Kotlin, а Byods [7] є прикладом вбудовування системи Datalog у Rust як процедурного макросу.

Однією з поширених проблем вбудовування мови є різниця між логічною та основною моделями об’єктів. У логіці ми зазвичай працюємо з неінтерпретованими функціями, тоді як нетипізоване представлення є рідкістю в об’єктно-орієнтованих мовах зі статистично типізованими функціями. Інша проблема полягає в організації двонаправленого обміну даних під час виконання логічних програм. Логічна части-

на повинна мати можливість викликати імперативну частину і навпаки.

У даній роботі представлено структуру для вбудовування методів логічного програмування та програмування в обмеженнях у мову Scala шляхом побудови логічної DSL навколо уніфікації типізованої логіки.

1. Опис логічної структури

У даному розділі неформально описано частину необхідної логічної структури, яку потрібно вбудувати.

Базовими об'єктами в логічному програмуванні є логічні терми T, які складаються з:

- констант $c \in C$;
- логічних змінних $t \in T_i$;
- неінтерпретованих функцій $f_i(x_1,...,x_n) \in \Theta$.

Якщо у нас є певна концепція уніфікації термів (яка буде описана пізніше), ми можемо побудувати логічний вираз над термами E. Логічні вирази будуються як:

- функціональні терми $f_i(x_1,...,x_n)$, що представляють логічне значення $\Theta \subset E$;
- $True$ і $False$ є логічними константами, $True \in E$, $False \in E$;
- для $x \in E$, $y \in E$ ми можемо побудувати звичайну логічну операцію: кон'юнкція ($x \wedge y$), диз'юнкція ($x \vee y$);
- заперечення ($\neg x$) відіграє особливу роль і доступне не у всіх інтерпретаціях.

Тепер, коли у нас є логічний вираз, ми можемо побудувати логічну програму, що складається з правил, які описують факти, залежності між правилами та стратегію логічного пошуку.

Програма повинна мати форму, що залежить від основного логічного механізму. Наприклад, якщо ми хочемо емулювати семантику Prolog з алгоритмом інтерпретації SLDNF (Selective Linear Definite clause resolution with Negation as Failure — вибіркова лінійна резолюція з визначеними твердженнями із запереченням як відмовою) [8], кожен оператор має бути диз'юнктом Горна ($x :- y$), де ліва частина має бути функціональним символом або *True*, а права частина має бути кон'юнкцією логічних виразів. Крім того, диз'юнкти Горна в Пролозі можуть містити псевдотерм *cut*:

$$(x :- y_1 ... y_n) : \in R_{Prolog} \; -(x \in \Theta \wedge x = True), \quad y_i \in E \wedge y_i = cut.$$

Для Datalog у нас є диз'юнкт Горна без заперечення:

$$(x :- y_1 ... y_n) : \in R_{Datalog} \; -(x \in \Theta \wedge x = True), \quad y_i \in E \wedge y_i \neq \neg z.$$

Для програмування логіки обмежень наші функціональні символи мають бути простими фактами або виразами із сигнатур підтримуваних теорій.

2. Вбудовування логічних конструкцій у мову Scala

Далі опишемо вбудовування вищерозглянутих конструкцій у Scala. Перше питання полягає в тому, як побудувати універсальне відображення між логічними термами та об'єктами Scala, визначеними безпечним для типів способом.

Логічні терми та вирази матимуть два представлення Scala — для часу компіляції, яке розробник використовує для створення логічних програм, і часу виконання, що застосовується, коли логічний механізм викликає засоби низького рівня.

Почнемо з часу компіляції. Константи та неінтерпретовані функції (факти) відображаються в case-класи Scala. Терми з логічними змінними представлені за допомогою вбудовування HOAS [9]: представлення складного терму $f(x,y)$ є функціональним виразом Scala $(x : A, y : B) \Rightarrow f(x,y)$. Для логічних виразів використовуються відповідні оператори Scala: & або $\wedge$ для кон'юнкції; | або $\vee$ для диз'юнкції; ! для заперечення; |- для диз'юнкта Горна (|- обрано замість :- через нижчий пріоритет оператора).

Нижче наведено приклад такого вбудованого виразу:

```
case class Edge(x: Int, y: Int)
           derives Fact
case class Path(x: Int, y: Int)
           derives Fact

val transitiveClosure = Prolog {
   (x: Int, y: Int, z: Int) => {
     Path(x, y) |-  Edge(x, y)
     Path(x, z) |-  Path(x, y) &
     Edge(y, z)
   }
}
```

Тут Edge і Path — case-класи Scala. Ми можемо визначити набір фактів за допомогою того самого синтаксису,

```
val facts = Prolog{
    Path(1,2) |- true
    Path(2,3) |- true
    Path(3,4) |- true
}
```

або створити диз'юнкти Горна з колекції case-класів Scala:

```
val facts2 = Prolog.asFacts(Path(1,2),
                Path(2,3), Path(3,4))
```

Як бачимо, вбудовування зберігає коректність типізації: введення всередині диз'юнкта Горна перевіряється синтаксичним макросом, а введення Scala-констант — компілятором Scala.

Розробник програми може отримати результат, ініціалізувавши логічний інтерпретатор, що розуміє Prolog і виконує запит:

```
val results = Prolog.defaultEndinge(
   rules ++ facts).query(x => Path(1, x))
```

Низькорівневе вбудовування використовується, коли нам потрібно впровадити нашу власну поведінку уніфікації в логічну програму. Наприклад, у реальному житті факти часто потрібно отримувати зі сховищ ключ-значення або якоїсь документо-орієнтованої чи реляційної бази даних. API низького рівня дозволяє написати терм для запиту до бази даних під час інтерпретації та повернення набору даних як набору фактів у процесі інтерпретації.

Щоб написати такі плагіни, розробники повинні працювати з представленням логічних термів під час виконання.

Почнемо з логічної змінної:

```
case class LogicalVariable(id: Long,
           pos: LogicalVariablePosition)
           extends LogicalTerm
```

Ми припускаємо, що логічний механізм підтримує зв'язки між логічними змінними та ідентифікаторами, тому змінні мають лише тип даних long із доданою інформацією для налагодження.

Типізовані терми представлені об'єктом Scala, для якого можна описати алгоритм уніфікації:

```
sealed trait TypedLogicalTerm[T]
             extends LogicalTerm

case class LogicalConstant[
           T: Unificable](value: T)
           extends TypedLogicalTerm[T]

case class LogicalFunctionSymbol[T](
           unificator: Unifiable[T],
           args: IndexedSeq[LogicalTerm])
           extends TypedLogicalTerm[T]
```

Щоб вивести власний тип термів за допомогою власної уніфікації, розробнику потрібно визначити тип Scala та надати для нього реалізацію типокласу Unificable.

Сигнатура уніфікованого типокласу є такою:

```
trait Unifiable[T]  {
   def unify[R[_]:UnificationEnvironment,
             D]
       (value: T,
        term: LogicalTerm, ,
        bindings: Bindings)
        (using EngineInstanceData[D]):
        R[T]
}
```

Тут Bindings — це базована (ground) підстановка, що підтримується логічним механізмом, який надає звичайні методи роботи з підстановками.

```
trait Bindings {

   def get(logicalVariable:
       LogicalVariable):
       Option[LogicalTerm]

   def updated(logicalVariable:
       LogicalVariable,
       term: LogicalTerm): Bindings
```

```
   def bind[R[_],A,D]
       (variable: LogicalVariable,
        a:A)
       using unifiable: Unifiable[A],
       EngineData[D]): R[A]

   def bindM[R[_],A]
       (variable: LogicalVariable,
        a:R[A])
       (using unifiable: Unifiable[A]):
       R[A]

   def newVariable(): LogicalVariable
}
```

Оскільки різні механізми уніфікації мають різні можливості, тип повернення уніфікації загортається в тип R[_].

Це дає нам загальну сигнатуру для уніфікації, яку можна переносити між різними логічними механізмами. Зауважимо, що загалом різні механізми мають різну семантику уніфікації, і тип результату уніфікації також може відрізнятися.

Мінімальна уніфікація над логічними термами x і y, $unify(x, y)$, може бути описана як отримання заміни θ такої, що $x\theta = y\theta = mgu(x, y)$, де є мінімальний основний уніфікатор x і y, якщо $mgu(x, y)$ існує, інакше — невдача.

У логічному програмуванні в обмеженнях уніфікація може повертати набір можливих замін під час обробки логічної змінної, щоб ініціювати пошук у кінцевій області. Наприклад, $unify(x \le 5, x > 3)$ може повернути дві підстановки — $\{x = 4\}$ і $\{x = 5\}$.

ISO Prolog визначає можливість затримки для керування порядком кон'юнктивних тверджень, і одним із можливих результатів уніфікації може бути затримка уніфікації до моменту, коли вирішаться деякі залежності.

Ще одним аспектом є асинхронне виконання проти синхронного. Наприклад, у деяких промислових випадках краще мати уніфікацію, яка повертає набір можливих замін як асинхронний потік. Це дозволяє програмі негайно почати обробку результатів у конвеєрі, не чекаючи закінчення процесу уніфікації. Зауважимо, що цей тип асинхронного результату не можна представити за допомогою типу даних синхронного повернення.

UnificationEnvironment в Scala описує загальні можливості уніфікації, такі як синхронні чи асинхронні результати, можливість відкладати уніфікацію, доки не буде вирішено певний терм, здатність скорочувати або розгалужуватися після успіху тощо. Ми маємо відповідний клас типу, успадкований від UnificationEnvironment для кожного набору можливостей.

Уніфікація, задана користувачем, може створювати підстановки вручну або отримувати середовище з новими підстановками завдяки операціям прив'язки. Робота середовища уніфікації полягає у збереженні поточної підстановки.

Мінімальне середовище передбачає наступні методи формування результатів:

```
def success[A](a: A, bindings: Bindings):
    R[A]

def failure[A](a: A, right: LogicalTerm,
               reason): R[A]

def error[A](e: Throwable): R[A]
```

Також R[_] утворює монаду, де значення є функціями від поточних прив'язок до набору можливих майбутніх прив'язок. Тобто R[A] можна розуміти як набір усіх можливих зв'язків, на яких логічний вираз, відображений у Scala типу A, є істинним. Тому ми можемо використовувати існуючі методи для роботи з монадичними конструкціями, такі як dotty-cps-async [10].

MultiunificationContext додає можливість повертати більше, ніж один результат уніфікації. Коли підтримується мультиуніфікація, ми можемо використовувати диз'юнкції

```
def or[A](x:R[A]*): R[A]
```

Проілюструємо використання уніфікації, заданої користувачем, написавши уніфікацію для класу, що виконує запит до бази даних документів. Припустимо, що у нас є клас User:

```
case class User(id: Long, name: String,
                email: String,
                phoneNumber: String,
                score: Int)
```

Ми хочемо написати низькорівневий механізм уніфікації так, аби перевірка

фактів користувача запускала запити з бази даних. Для цього визначимо даний Unifiable для нашого користувача як показано на рис. 1. З'єднання з базою даних передається для уніфікації з механізму в дані екземпляра.

Логіка уніфікації проста — якщо ми бачимо незв'язане логічне значення, тоді отримуємо всіх користувачів і надсилаємо в систему нові зв'язки з кожним користувачем за допомогою уніфікатора фактів. (Тобто, коли система запускатиме уніфікацію над екземпляром, повернутим із бази даних, буде викликано userFact unify). На константі здійснюватиметься перевірка, чи присутня ця константа в нашій базі даних, і якщо ми маємо частково заповнений вираз, створюватимуться запит до бази даних і нові зв'язки для кожного результату.

Для виконаня цього запиту, ми повинні встановити для ConnectionProvider значення instanceData

```
val results =
   Prolog.defaultEndinge(rules).
          withData(connectionProvider).
          query(x => ...)
```

Зауважимо, що цей приклад усе ще нереалістичний: отримання всіх баз даних без індексування рідко може бути прийнятним вибором.

Щоб подолати це, функція DelayedUnification дозволяє контролювати порядок розпізнавання логічних змінних у прив'язках.

API виглядає таким чином:

```
trait DelayUnificationContext[R[_]]
      extends UnificationContext[R] {

   def waitResolved(v: LogicalTerm):
       R[LogicalTerm]
}
```

Ця операція перевіряє, чи є поточний терм логічною змінною. Якщо ж так, вона призупиняє виконання поточного уніфікованого терму та перемикає логічну машину на обробку інших виразів, розташованих у тому ж диз'юнкті Горна. Якщо цей терм отримує значення або в поточному диз'юнкті Горна не залишилося цілей, виконання уніфікації відновлюється.

```
  object User {

   val userFact = Fact.derived[User]

   given Unifiable[User] with
     override def unify[R[_] : UnificationEnvironment, ConnectionProvider](
           value: UserInDB, term: LogicalTerm, bindings:Bindings)(
              using EngineInstanceData[ConnectionProvider]): R[User] = {
       val connection = summon[EngineInstanceData[ConnectionProvider]].get.connection
       term match
         case lv:LogicalVariable =>
           // retrieve all the users from the database and
           val users:List[User] = connection.collection[User]("users").find()
           or(users.map(u=>bindings.bind(lv,u)(using userFact)):_*)
         case lc@LogicalConstant(value) =>
           connection.collection[User]("users").findOne({ "id" -> value }) match
             case Some(userInDb) => userFact.unify[R](userInDb, lc, bindings)
             case None => failure(value,term)
         case LogicalFunctionSymbol(unifiable, args) =>
           if (unifiable.checkType[User]) then
             val query = Map.empty ++ args(0).ground.map{"id" -> id }
                                   ++ args(1).ground.map{ "name" -> name }
                                   ++ args(2).ground.map{ "email" -> email }
                                   ++ args(3).ground.map{ "phoneNumber" -> phoneNumber }
             val users = connection.collection[User]("users").find(query)
             or(users.map(u=>bindings.bind(lv,u)(using userFact)):_*)
           else
             failure(value,term)
     }
  }
```

Рис. 1. Визначення Unifiable для користувача

Тому, якщо ми хочемо уніфікувати роботу користувача лише тоді, коли деякі з індексованих властивостей вирішено, ми можемо змінити наш приклад на

```
override def unify[R[_] :
  (UnificationEnvironment &&
   DelayUnificationEnvironment),
  ConnectionProvider](
    value: UserInDB,
    term: LogicalTerm,
    bindings:Bindings)(
      using EngineInstanceData
           [ConnectionProvider]): R[User]
      = reify[R]{
        waitResolved(term) match
        case lv:LogicalVariable =>
          error(RuntimeException(
                "Attempt to query all
                 users"))
        case lc@LogicalConstant(value) =>
          ... // the same logic as in
              // the previous example
     }
```

Тепер запит, який вимагатиме отримання всіх об'єктів із бази даних, не вдасться виконати. Розглянемо також складену сигнатуру класу типів для unify, && визначений як

```
type &&[A[_[_]],B[_[_]]] =
              [R[_]] =>> A[R] & B[R]

R[_]: MultyUnificationEnvironment &&
      DelayUnificationEnvironment
```

Це дозволить використовувати unify у всіх середовищах, які підтримують як multi, так і delay.

API DelayUnificationEnvironment підходить для випадків, коли факти користувача потрібні для доступу до деталей про інші події. Але що, як ми дійсно хочемо отримати довгий список результатів із бази даних?

AsyncUnificationEnvironment додає до можливостей взаємодію з асинхронними обчисленнями в монадній упаковці. Це додавання наступних методів:

```
def adoptAsync[A](value: F[A]): R[A]

def asyncResult[[A](output:
                    AsyncList[F,A]): R[A]
```

Тип F[_] — це наша асинхронна монада, яка визначається як псевдонім абстрактного типу в AsyncUnificationEnvironment, для уникнення додавання F[_] до сигнатури unify.

Робота першого методу полягає в прийнятті результату асинхронного обчислення в поточну монаду. Наявність цього методу дозволяє нам реалізувати клас типів для перетворення між монадами та використовувати reflect[F] в дужках reify[R].

Другий метод — це вихід асинхронного потоку результатів, який буде асинхронно оброблений логічною системою як набір можливих альтернатив. Розробнику слід бути обережним із отриманням потоків у базі даних, оскільки на стороні клієнта легко виконувати неефективні об'єднання. У більшості випадків краще показати, що факт витягується з баз даних основною мовою. Код, який визначає факт QueryLanguage, може виглядати, як показано на рис. 2.

```
case class User(id: Long, name: String, email: String, phoneNumber: String,
                score: Int) derives Fact

case class QueryAllUsers(u: User)

object QueryAllUsers {
 given Unifiable[QueryAllUsers] with
   override def unify[R[_] : AsyncUnificationEnvironment.Aux[IO],ConnectionProvider](
     value: UserInDB, term: LogicalTerm, bindings: Bindings,
     instanceData: InstanceData[ConnectionProvider]): R[User] = reify[R] {
     val collection = instanceData.get.connection.collection[User]("users")
     term match
       case lv: LogicalVariable =>
         reflect(collection.asyncFind().map(u => bindings.bind(lv, QueryUser(u)))
       case lc@LogicalConstant(value) =>
         ... // logic similar to the previous example
   }
 }
}
```

Рис. 2. Визначення факту QueryLanguage

Тепер ми можемо використовувати QueryAllUsers у логічній програмі та писати такі логічні правила:

```
AllUsersAreScored |- QueryAllUsers(u) &
                     AssignScore(u)
```

Приклад усе ще спрощений, наступним природним кроком є розширення нашої здатності вбудовування для перекладу логічного виразу в запити до бази даних. Цього можна досягти напрочуд легко, оскільки побудова DSL-запитів у Scala, яка відображається в набір класів, є популярною технікою. Усе, що нам потрібно зробити, — це реалізувати Unificable для класу запитів обраної бібліотеки доступу до бази даних.

Висновки

Запропонована структура для вбудовування сімейства мов декларативної логіки в Scala із уніфікацією на основі монад, що може бути перепрограмована користувачем. Відмінності в можливостях логічних механізмів можна виразити як підтипи монад уніфікації. Такий спосіб дає змогу генерувати одну реалізацію налаштовуваної уніфікації для вбудовування різних логічних систем у Scala та використовувати вбудовані сторонні проблемно-орієнтовані мови у логічних виразах. Монадичний API, можливо, з використанням прямого стилю поверх нього, надає розробнику програми простий та інтуїтивно зрозумілий інструмент для реалізації власної логіки всередині уніфікації.

Монадичне представлення логічного обчислення є добре відомою технікою. Стандартна бібліотека Haskell включає трансформатор монад LogKit [11] на основі [12]. У світі функціонального програмування засоби логічного програмування часто вважаються невеликими доповненнями до існуючих засобів мови. Як бачимо, більшість логічних фреймворків у функціональній мові реалізують представлення алгоритмів «генерування та фільтрування», які мають обмежене практичне застосування, тому що справжня сила логічного програмування часто розкривається з можливістю складної оптимізації, такої як індексування термів затримки цілі або розв'язувачі, що задаються користувачем [8, 13].

Для Scala існує низка різних реалізацій вбудовування логічного програмування, серед яких перенесення монади Haskell LogKit [11], Scalano [14], які представляють логічний зв'язок як об'єкт Scala над типізованими термами зі зворотним обчислювачем, Fusemate [15], який поєднує логіку першого порядку зі спеціалізованою мовою для аналізу та моделювання систем, заснованих на подіях.

Наш підхід забезпечує чітке відділення логічної дедукції: код Scala використовується лише для уніфікації, що задана користувачем, але загальне виконання цілі є зовнішньою інтерпретацією, яка може реалізувати різні стратегії. Такий спосіб надає модульність і гарну інтеграцію з рештою екосистеми, але водночас існують недоліки. В ситуації, коли API є синхронним і дані знаходяться в пам'яті, монадичний інтерфейс може бути вузьким місцем, що буде перешкоджати досягненню високої продуктивності.

Подальша робота полягає в розширенні набору підтримуваних алгоритмів і розв'язувачів для природного інтегрованого програмування в обмеженнях. Користувацькі уніфікації також можуть бути елементом системи переписування правил (в TermWare також використовується уніфікація, що може бути перепрограмована користувачем), тому додавання підтримки програмування на основі правил також може бути вектором розширення.

Література

1. The Flix Programming Language [Електронний ресурс]. Доступ до ресурсу: https://flix.dev (дата звернення: 27.11.2023).
2. Magnus M., Starup J.L., Lhoták O. Flix: a meta programming language for Datalog. Datalog. *4th International Workshop on the Resurgence of Datalog in Academia and Industry (Datalog 2.0 2022)*. 2022. P. 202–206. [Електронний ресурс]. Доступ до ресурсу: https://ceur-ws.org/Vol-3203/short8.pdf (дата звернення: 27.11.2023).
3. Logtalk [Електронний ресурс]. Доступ до ресурсу: https://logtalk.org (дата звернення: 27.11.2023).

4. Doroshenko A., Shevchenko R. A rewriting framework for rule-based programming dynamic applications. *Fundamenta Informaticae*. 2006. Vol. 72, N 1–3. P. 95–108.
5. Мамедов Т.А., Дорошенко А.Ю., Шевченко Р.С. Засіб статичного аналізу .NET програм за допомогою переписувальних правил. *Проблеми програмування*. 2020. № 2–3. С. 157–163.
6. Ciatto G., Calegari R., Omicini A. 2P-Kt: a logic-based ecosystem for symbolic AI. *SoftwareX*. 2021. Vol. 16, 100817. P. 1–7. [Електронний ресурс]. Доступ до ресурсу: https://doi.org/10.1016/j.softx.2021.100817 (дата звернення: 27.11.2023).
7. Sahebolamri A., Barrett L., Moore S., Micinski K. Bring your own data structures to Datalog. *Proceedings of the ACM on Programming Languages*. 2023. Vol. 7, OOPSLA2, Article 264. P. 1198–1223. [Електронний ресурс]. Доступ до ресурсу: https://doi.org/10.1145/3622840 (дата звернення: 27.11.2023).
8. Sterling L., Shapiro E. The art of Prolog advanced programming techniques. 2nd ed. Cambridge, MA, USA: MIT Press, 1994. 560 p.
9. Pfenning F., Elliott C. Higher-order abstract syntax. *ACM SIGPLAN 1988 conference on Programming language design and implementation (PLDI'88). ACM SIGPLAN Notices*. 1988. Vol. 23, N 7. P. 199–208. [Електронний ресурс]. Доступ до ресурсу: https://doi.org/10.1145/960116.54010 (дата звернення: 27.11.2023).
10. Shevchenko R. Project paper: embedding generic monadic transformer into Scala. *In Swierstra W., Wu N. (eds) Trends in Functional Programming. TFP 2022. Lecture Notes in Computer Science*. Vol. 13401. Cham: Springer, 2022. P. 1–17. [Електронний ресурс]. Доступ до ресурсу: https://doi.org/10.1007/978-3-031-21314-4_1 (дата звернення: 27.11.2023).
11. LogKit [Електронний ресурс]. Доступ до ресурсу: https://github.com/logkit/logkit (дата звернення: 27.11.2023).
12. Kiselyov O., Shan C., Friedman D.P., Sabry A. Backtracking, interleaving, and terminating monad transformers (functional pearl). *10th ACM SIGPLAN international conference on Functional programming (ICFP'05)*. 2005. New York: ACM, 2005. P. 192–203. [Електронний ресурс]. Доступ до ресурсу: https://doi.org/10.1145/1086365.1086390 (дата звернення: 27.11.2023).
13. Körner P., Leuschel M., Barbosa J., Costa V. et al. Fifty years of Prolog and beyond. *Theory and Practice of Logic Programming*. 2022. Vol. 22, N 6. P. 776–858. [Електронний ресурс]. Доступ до ресурсу: https://doi.org/10.48550/ arXiv.2201.10816 (дата звернення: 27.11.2023).
14. Amin N., Byrd W.E., Rompf T. Lightweight functional logic meta-programming. *In Lin A. (eds) Programming Languages and Systems. APLAS 2019. Lecture Notes in Computer Science*. Vol. 11893. Cham: Springer, 2019. [Електронний ресурс]. Доступ до ресурсу: https://doi.org/10.1007/978-3-030-34175-6_12 (дата звернення: 27.11.2023).
15. Baumgartner P. The Fusemate logic programming system. *28th International Conference on Automated Deduction (Automated Deduction – CADE 28)*. Berlin: Springer, 2021. P. 589–601. [Електронний ресурс]. Доступ до ресурсу: https://doi.org/10.1007/978-3-030-79876-5_34 (дата звернення: 27.11.2023).

References

1. The Flix Programming Language [Online]. Available from: https://flix.dev [Accessed 27/11/2023].
2. Magnus, M., Starup, J.L. & Lhoták, O. (2022) Flix: a meta programming language for Datalog. Datalog. *4th International Workshop on the Resurgence of Datalog in Academia and Industry (Datalog 2.0 2022)*. p. 202-206. [Online]. Available from: https://ceur-ws.org/Vol-3203/short8.pdf [Accessed 27/11/2023].
3. Logtalk [Online]. Available from: https://logtalk.org [Accessed 27/11/2023].
4. Doroshenko, A. & Shevchenko, R. (2006) A rewriting framework for rule-based programming dynamic applications. *Fundamenta Informaticae*. 72 (1-3). p. 95-108.
5. Mamedov, T.A., Doroshenko, A.Yu. & Shevchenko, R.S. (2020) Static analysis of .NET programs using rewriting rules. *Problems in programming*. (2–3). p. 157-163. (in Ukrainian)
6. Ciatto, G., Calegari, R. & Omicini, A. (2021) 2P-Kt: a logic-based ecosystem for symbolic AI. *SoftwareX*. 16, 100817. p. 1-7. [Online]. Available from: https://doi.org/10.1016/j.softx.2021.100817 [Accessed 27/11/2023].
7. Sahebolamri, A. et al. (2023) Bring your own data structures to Datalog. *Proceedings of the*

ACM on Programming Languages. 7 (OOPSLA2), Article 264. p. 1198-1223. [Online]. Available from: https://doi.org/10.1145/3622840 [Accessed 27/11/2023].
8. Sterling, L. & Shapiro, E. (1994) The art of Prolog advanced programming techniques. 2nd ed. Cambridge, MA, USA: MIT Press.
9. Pfenning, F. & Elliott, C. (1988) Higher-order abstract syntax. *ACM SIGPLAN 1988 conference on Programming language design and implementation (PLDI'88). ACM SIGPLAN Notices*. 23 (7). p. 199-208. [Online]. Available from: https://doi.org/10.1145/960116.54010 [Accessed 27/11/2023].
10. Shevchenko, R. (2022) Project paper: Embedding generic monadic transformer into Scala. *In Swierstra, W., Wu, N. (eds.) Trends in Functional Programming. TFP 2022. Lecture Notes in Computer Science*. 13401. p. 1-17. Cham: Springer. [Online]. Available from: https://doi.org/10.1007/978-3-031-21314-4_1 [Accessed 27/11/2023].
11. LogKit [Online]. Available from: https://github.com/logkit/logkit [Accessed 27/11/2023].
12. Kiselyov, O. at el. (2005) Backtracking, interleaving, and terminating monad transformers (functional pearl). *10th ACM SIGPLAN international conference on Functional programming (ICFP'05)*. New York: ACM. p. 192-203. [Online]. Available from: https://doi.org/10.1145/1086365. 1086390 [Accessed 27/11/2023].
13. Körner, P. et al. (2022) Fifty years of Prolog and beyond. *Theory and Practice of Logic Programming*. 22 (6). p. 776-858. [Online]. Available from: https://doi.org/10.48550/arXiv.2201.10816 [Accessed 27/11/2023].
14. Amin, N., Byrd, W.E. & Rompf, T. (2019) Lightweight functional logic meta-programming. *In Lin, A. (eds.) Programming Languages and Systems. APLAS 2019. Lecture Notes in Computer Science*. 11893. Cham: Springer. [Online]. Available from: https://doi.org/10.1007/978-3-030-34175-6_12 [Accessed 27/11/2023].
15. Baumgartner, P. (2021) The Fusemate logic programming system. *28th International Conference on Automated Deduction (Automated Deduction – CADE 28)*. Berlin: Springer. p. 589-601. [Online]. Available from: https://doi.org/10.1007/978-3-030-79876-5_34 [Accessed 27/11/2023].

Одержано: 30.11.2023

***Про авторів*:**

Шевченко Руслан Сергійович,
кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість наукових публікацій
в українських виданнях – 11.
Кількість наукових публікацій
в зарубіжних виданнях – 8.
http://orcid.org/0000-0002-1554-2019,

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень, професор
кафедри інформаційних систем
та технологій Національного Технічного
Університету України
«КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – понад 200.
Кількість наукових публікацій
в зарубіжних виданнях – понад 80.
Індекс Гірша – 7.
http://orcid.org/0000-0002-8435-1451,

Яценко Олена Анатоліївна,
кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість публікацій
в українських виданнях – 60.
Кількість зарубіжних публікацій – 21.
http://orcid.org/0000-0002-4700-6704.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 60 33.
E-mail: ruslan@shevchenko.kiev.ua,
doroshenkoanatoliy2@gmail.com,
oayat@ukr.net

УДК 004.4 http://doi.org/10.15407/pp2024.01.012

Д.О. Бухаленков, Т.М. Заболотня

МОДИФІКОВАНИЙ МЕТОД ПОШУКУ КЛЮЧОВИХ СЛІВ ТА ТЕРМІНІВ У ТЕКСТОВИХ ДАНИХ

У даній статті розглядається питання автоматизованого пошуку ключових слів та термінів у текстових даних. Для підвищення ефективності засобів автоматизованого пошуку ключових слів у тексті за критеріями абсолютної точності та повноти за Жаккаром розроблено модифікацію одного з найсучасніших методів для пошуку ключових слів. Запропоновано модифікацію існуючого гібридного методу пошуку ключових слів, що враховує складні залежності між парами слів у тексті для визначення багатослівних виразів, що, на відміну від оригінального методу, дозволяє знаходити ключові терміни, які складаються з кількох слів. Здійснені випробування створеної модифікації гібридного методу пошуку ключових термінів показали ефективність її використання для пошуку ключових термінів у текстах у порівнянні з існуючими аналогами.

Ключові слова: ключові слова, ключові терміни, оброблення текстових даних, Python, стенфордська класифікація.

Вступ

Основний зміст будь-якого тексту можна описати одним або кількома словами з цього тексту, що відображають його суть. Такі слова називають ключовими словами. В більшості випадків кількість таких слів становить близько десяти [1]. Іноді ключовими можуть вважати не тільки слова, а й цілі словосполучення і речення.

Незважаючи на просте визначення поняття ключового слова, процес пошуку ключових слів є складним аналітичним завданням. Не існує ідеального способу визначення переліку ключових термінів для довільного тексту будь-якої тематики. Кожен текст має свою структуру, стиль викладення, стилістичні особливості написання [2].

Слід зазначити, що задача пошуку ключових слів виникає у багатьох сферах, пов'язаних з обробленням текстових даних. Так інформація про ключові слова використовується у інформаційному текстовому пошуку, класифікації, кластеризації даних. Водночас важливою вимогою до методів визначення ключових слів є можливість їх автоматизації, адже обсяги даних, які проходять через сучасні електронні пристрої та системи, неможливо ефективно обробляти вручну.

Упродовж багатьох років досліджень спеціалістами було запропоновано методи, різні за ефективністю та умовами застосування. Одні методи – добре налаштовані на оброблення текстів вузької тематики, зокрема, технічної літератури, інші – можуть бути методами ширшого застосування. Однак досі триває пошук шляхів покращення точності методів для пошуку ключових слів та підвищення ступеня їхньої універсальності щодо застосування до різних типів текстів.

Метою даної статті є підвищення ефективності засобів для автоматизованого пошуку ключових слів у тексті за критеріями абсолютної точності та повноти за Жаккаром шляхом модифікації одного з сучасних методів пошуку ключових слів та за допомогою використання сучасних лінгвістичних програмних пакетів.

Існуючі методи

Не існує ідеального "золотого правила", за яким можна було б визначити набір ключових слів для будь-якого тексту. Відомі методи можна умовно поділити на кілька основних груп:

– Статистичні методи – найстарішими можна назвати методи, що ґрунтуються на використанні статистичних даних, отриманих під час аналізу тексту [3]. Одними із перших статистичних закономірностей, виявлених для природномовних текстових даних, можна вважати здобутки

американського лінгвіста та економіста Джорджа Ципфа. Виведені ним закони про розподіл слів у текстах є основами багатьох відомих статистичних методів.

Статистичні методи в основному аналізують частоту входжень слів у тексті, їхню довжину, відстань у тексті між ними. Приклади відомих статистичних метрик та методів – метрика **TF-IDF**, методи виявлення **семантичного поля**, метод **системного зважування слів за частотою та довжиною**, метод **k-factor**, метод **C-value**, **TERMC** та інші.

Основними **перевагами** статистичних методів можна вважати їхню **швидкість** та **мовонезалежність**, адже статистичні закономірності майже однаково прослідковуються у більшості природних мов.

Характерні **недоліки** таких методів полягають у великому обсязі вербального шуму в отриманих результатах та недостатній точності у застосуванні на текстах невеликих розмірів.

– Словникові методи – методи, що використовують заздалегідь зібрані словникові дані, або тезауруси з деяких тематик [4]. На відміну від статистичних, такі методи здатні надавати **більш точні** результати з меншою кількістю вербального шуму. Однак для отримання точних результатів треба мати дуже докладні тезауруси з відповідної до тексту тематики. З цього випливає, що методи, побудовані на основі використання словників, складно застосовувати до текстів **вузької або нової тематики**.

– Гібридні методи – сучасні розробки використовують поєднання особливостей статистичних та словникових методів для найбільш ефективного пошуку ключових слів [5]. **Статистичні закономірності** допомагають швидко знайти основний масив потенційних ключових слів, а моделі машинного навчання, натреновані на словникових даних, **збільшують точність і відсіюють вербальний шум**, в результаті чого на виході отримується набір ключових слів. Серед популярних сучасних програмних інструментів для оброблення природномовних текстових даних: Python NLTK, Stanford NLP, Keras, spaCy тощо.

Обраний метод для модифікації

Гібридний метод пошуку ключових слів в англомовних текстах, що був запропонований українським фахівцем О.В. Яхимовичем [6], належить до останньої з трьох вищезазначених груп методів. Метод використовує інструменти сучасних програмних синтаксичних аналізаторів для оброблення текстів і отримання необхідних даних для подальшого зважування слів-кандидатів у ключові слова. Наведемо основні етапи методу:

1. Синтаксичний аналіз тексту і отримання даних про зв’язки між парами слів і частини мови, до яких належать слова тексту.
2. Фільтрування пар слів, зв’язки між якими належать до переліку неінформативних.
3. Заміна займенників у парах слів відповідними іменниками.
4. Відсіювання слів, які при синтаксичному аналізі було зараховано до неінформативних частин мови.
5. Фільтрування стоп-слів.
6. Визначення кількості зв’язків для кожного слова з пари.
7. Прийняття перших **n** слів з найбільшою кількістю зв’язків як ключових (де n - бажана кількість шуканих ключових слів).

Для отримання пар слів використовується **стенфордська класифікація** [7] зв'язків між лексичними одиницями речень тексту. Розробниками методу шляхом численних випробувань було визначено 7 типів зв’язків, що не несуть суттєвого змістовного навантаження і не відіграють важливої ролі в контексті пошуку ключових слів. Це зв’язки: CC, DET, EXPL, FIXED, PUNCT, REF, ROOT.

Для фільтрації слів, що належать до неінформативних частин мови, автори гібридного методу використовують **класифікацію Пенна** [8] і виділяють 21 тег з цієї класифікації: CC, CD, DT, EX, IN, LS, MD, PDT, POS, PRP, PRP$, RP, SYM, TO, UH, WDT, WP, WP$, WRB, -LRB-, -RRB-.

Заміна займенників на відповідні іменники відбувається за допомогою ана-

лізу кореференційних зв'язків між словами в тексті.

Розробники методу запевняють, що запропонований гібридний метод має приріст повноти у межах від 8,1% до 12,7% **за метрикою Жаккара**, та від 9,1% до 14,3% **абсолютної точності** пошуку ключових слів у порівнянні з існуючими аналогами. Отже, метод можна вважати одним із найбільш точних серед сучасних розробок, і його модифікація для отримання ще більш точних результатів вбачається перспективною.

Гіпотеза №1 про підвищення точності "Гібридного методу"

Головною особливістю оригінального методу можна вважати визначення кількості синтаксичних зв'язків між словами і відсіювання пар слів із неінформативними типами зв'язків та слів, що належать до неінформативних частин мови. Таким чином, на якість результатів пошуку ключових слів насамперед впливає вміст списків неінформативних типів зв'язків та частин мови, що були визначені авторами заздалегідь. Отже, **модифікація цих списків** потенційно може покращити кількісні характеристики якості отримуваних результатів.

Авторами даної статті було вирішено зробити наступні модифікації списків:

– виключити зі списку неінформативних частин мови тег **CD**, або **cardinal number** (кардинальне число). Було висунуте припущення, що важливі для змісту ключові слова або фрази можуть містити конкретні числа, як-от у виразі "Order **767**". Якщо є якийсь загальновідомий історичний наказ під номером 767, внесення такого числа до списку ключових слів покращить результати пошуку і ймовірність знаходження даного ресурсу, де описано цей наказ. За оригінальним методом числівник 767 було б повністю вилучено зі списку потенційних ключових слів, що, можливо б, погіршило якість інформаційного пошуку;

– вилучити зі списку неінформативних частин мови тег **RP**, або **particle** (частка). В деяких специфічних поняттях чи термінах може міститися слово-частка. Наприклад, у реченні "Located right on the airfield, guests can watch other planes take **off** and land." "off" є частиною "take off", тобто злітати. Якщо відфільтрувати цю частку зі списку ключових слів, то пошуковий запит, що містить "take off", не буде чітко відповідати набору ключових слів;

– включити до списку неінформативних типів зв'язків тип **ccomp**, або **clausal complement** (комплементна клаузальна конструкція). Найчастіше такий зв'язок трапляється між дієсловом або прикметником і додатком. Зв'язок є досить специфічним і часто не несе достатнього інформаційного змісту, щоб додавати ваги словам-кандидатам у ключові слова.

Були проведені випробування із запропонованими модифікаціями списків. Результати тестування на чисельних текстах тез статей з наукових журналів показали, що списки неінформативних частин мов та типів зв'язків, виділені авторами оригінального методу, є ефективними. Майже в усіх випадках результати не змінювалися, спостерігалися лише невеликі відхилення в той чи інший бік на одне слово. Середні значення абсолютної точності та повноти за Жаккаром виявилися майже однаковими (рис.1).

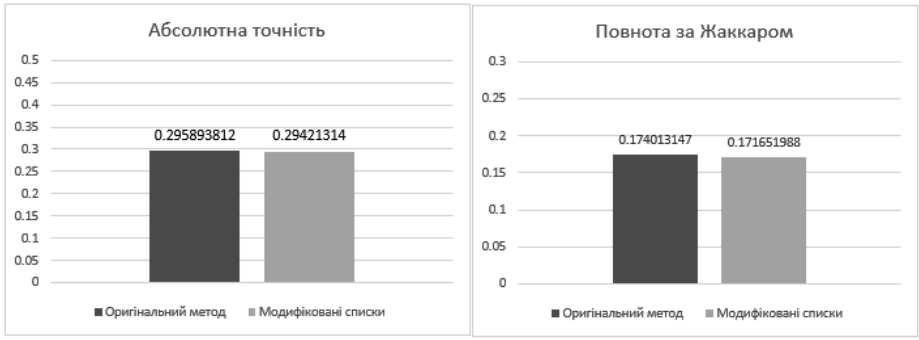

Рис.1. Результати тестування методу з модифікованими "неінформативними" списками

Отож, кількісні показники якості результатів роботи методу зі зміненими списками за абсолютною точністю знайдених ключових слів та повнотою Жаккара були практично однакові з результатами оригінального методу. Отже, перша гіпотеза була спростована.

Гіпотеза №2 про використання інформації про багатослівні вирази для пошуку ключових термінів

У оригінального методу є суттєвий недолік – він дозволяє шукати **лише однослівні ключові терміни**. Під час виконання алгоритму, що реалізує метод, пари слів роз'єднуються в окремі слова, після чого для кожного слова окремо визначається кількість зв'язків, отже, на виході можна отримати лише окремі ключові слова. Використання однослівних ключових термінів сприяє більш загальному пошуку, але недостатньо добре покриває специфічні і конкретні запити.

Авторами цієї статті було вирішено модифікувати метод таким чином, щоб додати можливість пошуку ключових термінів, що складаються з кількох слів.

Проаналізуємо, які типи зв'язків зустрічаються між словами багатослівних ключових термінів. Для отримання "еталонного" переліку ключових термінів у рамках даного дослідження було вирішено використовувати статті наукових журналів, де до кожної з них є наданий авторами набір ключових слів та виразів, який можна вважати довідковим для проведення порівнянь [9].

Для тексту анотації до статті [10] дослідимо зв'язки між словами ключового терміна "ammonium perchlorate" (рис.2).

compound(perchlorate-6, ammonium-5)
compound(perchlorate-25, ammonium-24)
compound(perchlorate-32, ammonium-31)

Рис.2. Зв'язки між словами ключового терміна "ammonium perchlorate" для тексту анотації до статті [10]

Аналізатор визначив, що такий ключовий термін можна відшукати в тексті за зв'язками типу compound. Згідно таблиці типів синтаксичних зв'язків **Universal Dependencies** (стенфордська класифікація) [11] тип зв'язку compound належить до категорії **MWE (Multiword Expressions)**, тобто багатослівних виразів. Розглянемо, які типи синтаксичних зв'язків містить у собі категорія MWE.

Fixed. Використовується для позначення спеціальних службових конструкцій, фіксованих виразів, зворотів. Слова, що мають зв'язок fixed, не мають зв'язків інших типів з іншими словами. Приклади таких конструкцій в англійській мові: *as well as*, *because of*, *rather than*. На рис.3 наведено приклади речень, де визначено зв'язки типу fixed [12].

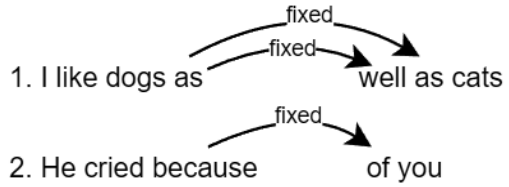

Рис. 3. Приклади речень зі зв'язком типу fixed

Flat. Цей тип зв'язку використовується для екзоцентричних виразів, тобто таких, де немає головного слова. До таких належать імена і дати. На рис.4 наведено приклади визначення зв'язків для імен.

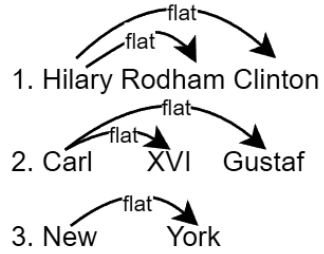

Рис. 4. Приклади визначення зв'язку flat для імен

Зв'язок flat також застосовується до виразів, де згадується титул або звання персони разом з ім'ям (рис.5).

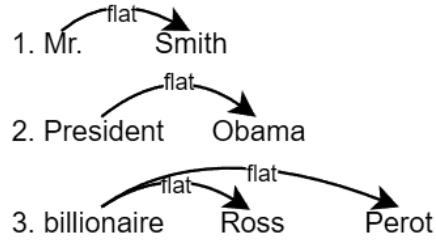

Рис. 5. Зв'язки flat для виразів з титулами або званнями

Для складених числових виразів також застосовується зв'язок flat [13]. На рис.6 наведено приклад виразів з датами або складними числівниками, де визначено цей тип зв'язку.

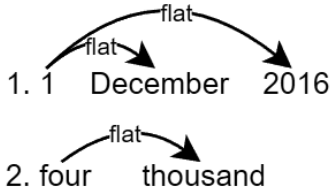

Рис. 6. Вирази з датами та складними числівниками, де визначено тип зв'язку flat

Compound. Цей тип зв'язку визначено у виразах ендоцентричного типу, де, на відміну від екзоцентричних, є головне слово. Такі вирази є сполученнями з кількох частин мови: іменникові сполучення, дієслівні, прикметникові, їхні комбінації та іноді серійні дієслівні конструкції. Більшою мірою виражені [14]:

– складними іменниками. Це можуть бути **Іменник** + **Іменник** (bus stop, fire-flies, football), **Прикментик** + **Іменник** (full moon, blackboard, software), **Дієслово** + **Іменник** (breakfast, washing machine, swimming pool), **Іменник** + **Дієслово** (sunrise, haircut) та інші сполучення (USB cell phone chargers);

– серійними дієслівними конструкціями. Синтаксична конструкція, де представлена послідовність двох або більше дієслів, які функціонують як один предикат та описують одну подію. В сучасній англійській мові майже не зустрічаються, але збереглися деякі вирази: let's go eat, come live with me.

– фразовими дієсловами. Комбінація дієслова і прийменника, або дієслова і прислівника, або одночасно дієслова і прийменника з прислівником, яка є окремим членом речення і утворює окрему семантичну одиницю. Найчастіше складаються із власне смислового дієслова та одного або декількох прийменників (рідше прислівників): keep on, pass out, look up, give up, put off, come across.

Декілька прикладів, наведених в довіднику Universal Dependencies, зображено на рис.7.

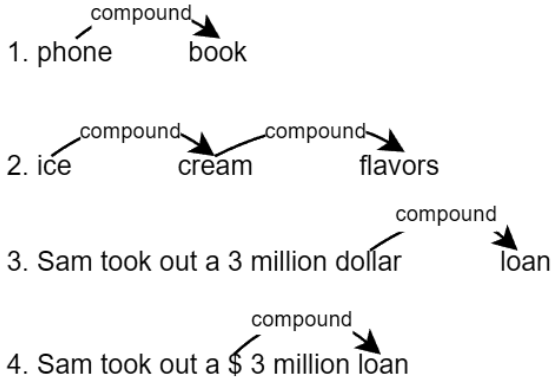

Рис. 7. Приклади виразів, де присутній зв'язок типу compound

Із перелічених трьох типів зв'язків fixed становить найменший інтерес з точки зору вирішення задачі пошуку ключових термінів. Fixed позначає спеціальні звороти і усталені сполучення слів, що утворилися історично або іншим шляхом, такі як instead of, rather than та інші. Вони слугують радше пов'язуючими функціональними частинами речень, але смислового навантаження самі собою не несуть.

Таким чином, з огляду на вищезазначене, для подальшого дослідження було обрано можливості використання знайдених зв'язків типу **flat** та **compound** для отримання списку ключових словосполучень із текстів.

Було знайдено наступні вирази: ['Honolulu', 'Hawaii'], ['Harvard', 'Law', 'Review'], ['Columbia', 'University'], ['Harvard', 'Law', 'School'], ['community', 'organizer'], ['law', 'degree']. Знайдені словосполучення дійсно певною мірою відображають зміст тексту, і теоретично підходять як ключові сліва.

Очевидно, що вибір з тексту всіх виразів з типами зв'язків flat та compound без якої-небудь фільтрації не є повноцінним рішенням задачі пошуку ключових виразів.

Як запевняють автори оригінального методу, його використання дозволяє досить точно знаходити окремі ключові слова [6]. Звідси можна використати визначений список окремих ключових слів для фільтрування отриманих багатослівних виразів. Адже логічно припустити, що якщо текст містить деякі ключові словосполучення або терміни з кількох слів,

окремі слова з них ймовірно будуть знайдені алгоритмом оригінального методу.

Таким чином, можна "відфільтрувати" всі знайдені багатослівні вирази, отримані з пошуку в тексті зв'язків flat та compound, використовуючи отримані раніше поодинокі ключові слова. Тобто, якщо в деякому знайденому з тексту виразі, що мав між словами зв'язки flat чи compound, міститься хоча б одне слово з набору ключових слів, такий вираз з великою ймовірністю буде ключовим для даного тексту. Проведені випробування на текстах тез статей наукових журналів підтверджують ефективність даного припущення за критеріями абсолютної точності та повноти пошуку ключових слів, про що йтиметься далі в цій статті.

Модифікований метод пошуку ключових слів та виразів

Із урахуванням вищенаведеного, пропонується модифікація оригінального методу з додатковими кроками для отримання списку ключових виразів та термінів до тексту. А саме:

– на етапі номер 1) оригінального методу збирається інформація про всі зв'язки типу flat і compound, з чого отримується набір усіх багатослівних виразів з такими зв'язками в тексті;

– після етапу номер 7) оригінального методу, враховуючи отримані результати, відбувається фільтрація отриманих на етапі 1) багатослівних виразів наступним чином: якщо ключове слово міститься у виразі, він потрапляє до списку ключових, інакше – відсіюється;

– в результаті отримується список ключових слів і список ключових виразів із двох або більше слів.

Список ключових виразів можна буде використовувати у процесі індексації ресурсів у пошукових системах, надаючи їм більший пріоритет збігу із пошуковим запитом користувача, адже ключові терміни точніше відображають зміст тексту, аніж поодинокі ключові слова.

Отже, в загальному вигляді запропонований авторами модифікований метод є таким:

1. Синтаксичний аналіз тексту і отримання даних про зв'язки між парами слів і частини мови, до яких належать слова тексту.
2. **Отримання з тексту набору всіх виразів з типами зв'язків flat та compound.**
3. Фільтрування пар слів, зв'язки між якими належать до переліку неінформативних.
4. Заміна займенників у парах слів відповідними іменниками.
5. Відсіювання слів, які під час синтаксичного аналізу було віднесено до неінформативних частин мови.
6. Фільтрування стоп-слів.
7. Визначення кількості зв'язків для кожного слова з пари.
8. Прийняття перших **n** слів з найбільшою кількістю зв'язків як ключові (де n - бажана кількість шуканих ключових слів).
9. **Фільтрація отриманих багатослівних виразів за допомогою попередньо отриманих ключових слів.**

На рис.8 наведено загальну схему запропонованого модифікованого методу.

Вибір метрик для визначення кількісних характеристик ефективності модифікованого методу

Враховуючи постановку задачі, а саме модифікацію методу з метою уможливлення отримання в результаті аналізу не тільки ключових слів, а й багатослівних ключових термінів та виразів, що може давати точніші за існуючі аналоги результати, ніж, час виконання нового модифікованого методу має не таке вирішальне значення, як точність і якість отриманих результатів. Отже, потрібно обрати метрики, що покажуть переваги використання модифікованого методу замість існуючих аналогів саме за знайденими ключовими словами та виразами.

Складно правильно оцінити точність знайдених ключових слів чи виразів для певного довільно взятого тексту, адже

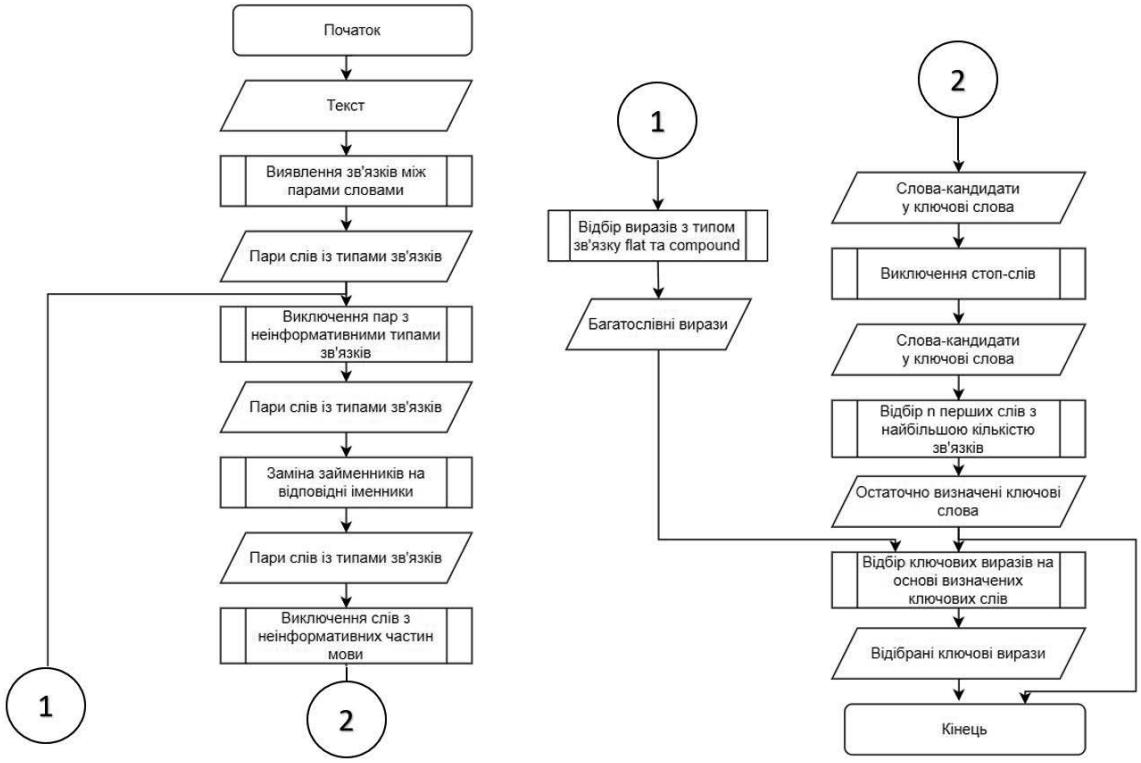

Рис.8. Схема етапів модифікованого методу для пошуку ключових виразів та термінів

хоча у ключових слів є деякі визначені характеристики, це доволі суб'єктивна оцінка. Тож різні спеціалісти можуть сперечатися щодо правильності визначення того чи іншого слова як ключового. Тому для досягнення максимально можливої об'єктивності було вирішено використовувати для випробування тексти із наперед визначеним набором ключових слів. Серед таких текстів є наукові статті, до яких при публікації в журналі автор сам підбирає набір ключових слів чи понять.

Для обґрунтування доцільності використання гібридного методу для пошуку ключових слів автори обрали дві метрики: **абсолютну точність** та **повноту за Жаккаром**. Коротко опишемо суть цих метрик.

Абсолютна точність визначається як відношення кількості правильно знайдених ключових слів за допомогою використання програмної реалізації методу до кількості ключових слів, визначених автором тексту.

Наприклад, якщо взяти множину еталонних ключових слів до тексту як A, а множину ключових слів, що було знайдено програмою як B, тоді абсолютну точність a пошуку ключових слів можна обчислити за формулою:

$$a = \frac{n(A \cap B)}{n(A)} \qquad (1)$$

де $n(A \cap B)$ – кількість правильно знайдених ключових слів; $n(A)$ – кількість еталонних ключових слів.

У свою чергу, **повнота за Жаккаром** визначається як відношення кількості правильно знайдених ключових слів до загальної кількості еталонних ключових слів і знайдених ключових слів мінус кількість правильно знайдених ключових слів. Повнота за Жаккаром J обчислюється за формулою:

$$J = \frac{n(A \cap B)}{n(A) + n(B) - n(A \cap B)} = \frac{n(A \cap B)}{n(A \cup B)} \qquad (2)$$

де $n(B)$ – кількість програмно знайдених ключових слів; $n(A \cup B)$ – кількість елементів об'єднання обох множин [16].

Ці дві метрики достатньо легко використати для порівняння двох множин окремих слів, адже тоді можна точно визначити, чи входить слово до переліку визначених автором, чи ні. Але з виразами і

ключовими термінами з кількох слів це складніше застосувати. Буде не зовсім справедливо порівнювати вирази один з одним, адже так втратиться велика кількість правильно знайдених ключових виразів, які лише трохи відрізняються. Наприклад, авторами статті [17] було визначено ключовий термін “reynolds number test”, а автоматизований метод знайшов вираз “reynolds number”. Порівнюючи вирази за логікою “один до одного” результат буде визначено помилковим, незважаючи на збіг 2 з 3 слів, і однаковий сенс двох виразів. Звідси постає питання, як порівняти два вирази із деяким порогом допустимої різниці складу виразів.

Є сфери, де науковці стикаються з подібною задачею оцінки точності збігу словесних виразів. До них можна віднести розроблення і тестування систем автоматичного розпізнавання мовлення або систем машинного перекладу. В обох випадках в результаті роботи систем на виході отримується текст або ж набір словесних виразів чи фраз, які треба порівняти з “еталонним” набором для оцінки якості роботи системи чи методу.

Однією із найвідоміших метрик для оцінювання роботи систем розпізнавання мовлення та машинного перекладу є **Word Error Rate (WER)**, або ж Частота Помилкових Слів [18]. Метрика базується на понятті відстані Левенштейна, але працює на рівні слів, а не символів. Визначається як відношення кількості замін слів, видалення слів, або додавання слів до довідкового варіанту для приведення його до вигляду отриманого автоматичною системою, до кількості слів у довідковому значенні.

Значення метрики **WER** може бути обчислене за наступною формулою:

$$WER = \frac{S + D + I}{N} \qquad (3)$$

де S – кількість замін; D – кількість видалень; I – кількість вставлень; N – кількість слів в “еталонному”, або довідковому варіанті.

Існує також і обернена до WER метрика – **WAcc**, або ж **Word Accuracy**, яка обчислюється як різниця одиниці і значення WER.

$$WAcc = 1 - WER \qquad (4)$$

Фактично це та сама метрика, що і WER, але кількісно відображається не помилка, а точність.

Експериментальним шляхом було вирішено встановити поріг значення WAcc, за якого вирази будуть вважатися рівними, у 66,66%. Із зниженням цього порогу спостерігалося збільшення вербального шуму в результатах пошуку, а із збільшенням – втрачалося більше ключових виразів, визначених авторами. Це означає, що за умови існування еталонного ключового виразу *reynolds number test*, знаходження алгоритмом виразу *reynolds number* буде вважатися успішним. Таким чином, ми можемо використовувати модифіковані метрики **абсолютної точності** та **повноти за Жаккаром** для оцінювання результатів роботи модифікованого методу.

Особливості програмної реалізації методу

Запропонований модифікований метод було реалізовано у вигляді консольного додатку на Python з можливістю взаємодії за CLI. Для виконання основних операцій оброблення природномовних текстів було використано платформу Python NLTK, а також допоміжний лінгвістичний пакет AllenNLP [19]. Для обчислення метрики WER використано пакет JiWER [20].

Використовуючи Python NLTK для пошуку зв’язків між парами лем в тексті рекомендується розглядати кожне речення окремо. Для цього необхідно розбити текст на речення, для чого використовується функція *sent_tokenize* з модуля *nltk.tokenize*.

Після того, як текст було розбито на речення, необхідно проаналізувати кожне і отримати список усіх зв’язків, або залежностей (dependency) між парами слів. Для цього застосовується модуль *StanfordDependencyParser*, а саме його метод *raw_parse*, який приймає на вхід речення в строковому вигляді, і на вихід віддає складну структуру-дерево з усіма зв’язками між парами слів.

Для отримання більш точних результатів за алгоритмом необхідно приводити слова до основної форми перед визначенням кількості зв'язків. Це прибере похибку результатів. У випадку, якщо одна й та сама сутність або поняття малися на увазі в різних частинах тексту, і вживалися в різних формах, вони будуть розцінені як різні слова при визначенні кількості зв'язків і зважуванні. Це може значно знизити вагу поняття в кінцевому випадку. Для приведення слів до основної форми використовується модуль *WordNetLemmatizer*, що імпортується з *nltk.stem*, а саме його метод *lemmatize*, який, однак, потребує дані про частину мови. Для цього використовується метод *pos_tag*, що дозволяє при надаванні тексту отримати для кожного слова інформацію, який тег частини мови має кожне слово з цього тексту.

Для фільтрації стоп-слів використовується словник стоп-слів, що надається модулем *stopwords* із набору *nltk.corpus*.

Для запуску додатку необхідно задати такі параметри за допомогою рядка CLI:

1. Шлях до файлу з вхідним текстом.
2. Шлях до файлу з результатами, що буде створено.
3. Кількість бажаних окремих ключових слів в результаті n. *Необов'язковий параметр*. У разі відсутності параметру за ключові будуть узяті слова-кандидати, що лишилися після кроків фільтрування.
4. Файл із переліком еталонних ключових слів, якщо такі є. Необов'язковий параметр. Використовується для перевірки точності роботи методу для тексту з наперед визначеними ключовими словами.

Випробування розробленого модифікованого методу

Програмну реалізацію розробленого модифікованого методу було протестовано на 50 довільних текстах тез до статей з наукових журналів [9]. Для порівняння було обрано існуючий сервіс для пошуку ключових слів **MonkeyLearn** [21], що є одним із найбільш популярних і ефективних. Розробники сервісу не розкривають, який саме метод пошуку ключових слів використовують, адже сервіс має багато платних функцій для аналізу контенту. Але згідно з інформацією про сервіс [22], використовується гібридний метод, що поєднує статистичні підходи та можливості машинного навчання.

Результати випробування у вигляді діаграм порівнянь середніх значень для метрик абсолютної точності пошуку ключових слів та повноти за Жаккаром зображено на рис.9.

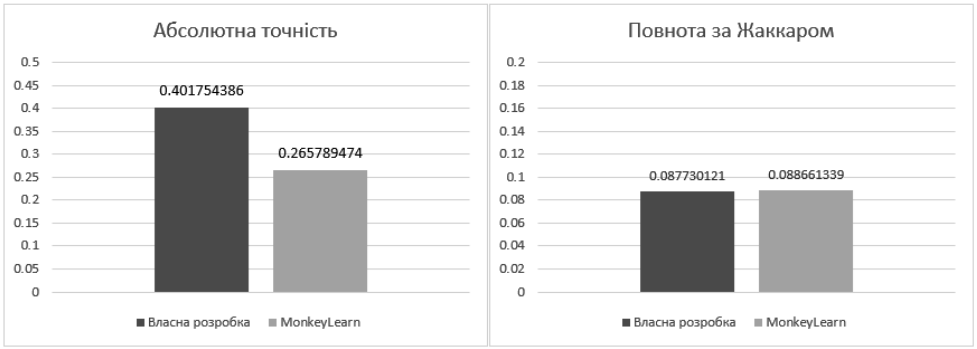

Рис.9. Результати випробування власної модифікації та сервіса MonkeyLearn

Середнє значення абсолютної точності для власної розробки – **0,402**, для сервісу MonkeyLearn – **0,266**, отже, власна розробка збільшує абсолютну точність пошуку ключових слів на **13,6%**. Середнє значення повноти за Жаккаром для власної розробки – **0,088**, для сервісу MonkeyLearn – **0,089**, отже маємо зменшення повноти пошуку ключових слів за Жаккаром на **0,1%**. Аналізуючи результати, можна стверджувати, що за невеликого зменшення повноти пошуку, модифікований метод має суттєве підвищення абсолютної точності порівняно з аналогом.

Висновки

У даній статті обґрунтовано актуальність проблеми пошуку ключових слів в тексті. Коротко описано та проаналізовано існуючі типи методів пошуку ключових слів, їхні переваги та недоліки.

Проаналізовано та обґрунтовано вибір для подальшого дослідження гібридного методу пошуку ключових слів за авторством О.В. Яхимовича. Висвітлено недоліки цього методу, та важливість їх уникнення. Висунуто гіпотези щодо підвищення ефективності методу та усунення недоліків.

На основі гіпотези про використання даних щодо багатослівних виразів у тексті для пошуку ключових термінів із кількох слів побудовано модифікацію оригінального методу. Це дозволяє шукати не лише окремі ключові слова, а й ключові терміни, що складаються з кількох слів.

Для випробування розробленого методу реалізовано програмне забезпечення у вигляді додатку мовою Python із використанням сучасних лінгвістичних програминих пакетів. Протестовано програмну реалізацію модифікованого гібридного методу на текстах тез статей із наукових журналів, отримані результати порівняно з результатами існуючого популярного сервісу MonkeyLearn.

Запропонована модифікація методу пошуку ключових слів збільшує абсолютну точність пошуку ключових слів у англомовних текстах із невеликим зменшенням повноти за Жаккаром.

Автори статті вважають за доцільне проведення подальших досліджень за такими напрямками:

–збільшення кількості випробувань на текстах різних розмірів та тематик;

– зменшення вербального шуму серед багатослівних ключових термінів, отриманих у результаті роботи методу;

– оформлення розробленого програмного забезпечення у вигляді простої для використання бібліотеки.

Література

1. Shibamouli Lahiri, Sagnik Ray Choudhury, Cornelia Caragea. Keyword and Keyphrase Extraction Using Centrality Measures on Collocation Networks, 2014.
2. H. M. Mahedi Hasan, Falguni Sanyal, Dipankar Chaki, Md. Haider Ali. An empirical study of important keyword extraction techniques from documents. 2017. In Proceedings of the 2017 1st International Conference on Intelligent Systems and Information Management, 91–94.
3. Rafael Geraldeli Rossi, Ricardo Marcondes Marcacini, Solange Oliveira Rezende. Analysis of Statistical Key-word Extraction Methods for Incremental Clustering. Proceedings of the 10th of the Encontro Nacional de Inteligˆencia Artificial e Computacional (ENIAC), Fortaleza, Brazil, 2013, 1–12.
4. Takashi Yamauchi, Dongshik Kang, Hayao Miyagi. The Keyword Search Using Thesaurus Concept, 2002 [Online] – Available from: https://koreascience.kr/article/CFKO200211921321260.pdf, last accessed 2024/01/08.
5. K. S. Sampada, N Kavya. Machine Learning Methods for Keyword extraction and Indexing, 2019.
6. Яхимович О.В., "Інформаційна технологія пошуку ключових слів на основі парсингу англомовних текстів", Вінниця, 2021.
7. Marie-Catherine de Marneffe, Christopher D. Manning (2008). Stanford typed dependencies manual [Online] – Available from: https://downloads.cs.stanford.edu/nlp/software/dependencies_manual.pdf, last accessed 2024/01/08.
8. Beatrice Santorini (1990). Part-of-Speech Tagging Guidelines for the Penn Treebank Project [Online] – Available from: https://www.cis.upenn.edu/~bies/manuals/tagguide.pdf, last accessed 2024/01/08.
9. Journal of Aerospace Technology and Management [Online] – Available from: https://jatm.com.br/jatm/issue/archive, last accessed 2024/01/08.
10. Rene Gonçalves, Koshun Iha, Francisco Machado, José Rocco. (2012). Ammonium Perchlorate and Ammonium Perchlorate-Hydroxyl Terminated Polybutadiene Simulated Combustion. Journal of Aerospace Technology and Management. 4.

11. Universal Dependency Relations [Online] – Available from: https://universaldependencies.org/u/dep/, last accessed 2024/01/08.
12. Fixed dependency [Online] – Available from: https://universaldependencies.org/u/dep/fixed.html, last accessed 2024/01/08.
13. Flat dependency [Online] – Available from: https://universaldependencies.org/u/dep/flat.html, last accessed 2024/01/08.
14. Compound dependency [Online] – Available from: https://universaldependencies.org/u/dep/compound, last accessed 2024/01/08.
15. Steven Bird, Ewan Klein, Edward Loper. (2009). Natural Language Processing with Python.
16. NC Chung, B. Miasojedow, M. Startek, A. Gambin (2019). "Jaccard/Tanimoto similarity test and estimation methods for biological presence-absence data". BMC Bioinformatics.
17. Maurício Silva, Victor Gamarra, Koldaev Vitor. (2009). Control of Reynolds number in a high speed wind tunnel. Journal of Aerospace Technology and Management. 1.
18. Dietrich Klakow, Peters Jochen (2002). "Testing the correlation of word error rate and perplexity". Speech Communication. 38 (1–2), 19–28.
19. AllenNLP Library [Online] – Available from: https://allenai.org/allennlp/software/allennlp-library, last accessed 2024/01/08.
20. JiWER [Online] – Available from: https://jitsi.github.io/jiwer/, last accessed 2024/01/08.
21. Keyword Extractor – MonkeyLearn [Online] – Available from: https://monkeylearn.com/keyword-extractor-online/, last accessed 2024/01/08.
22. Keyword Extraction: A Guide to Finding Keywords in Text – MonkeyLearn [Online] – Available from: https://monkeylearn.com/keyword-extraction/, last accessed 2024/01/08.

Одержано: 23.02.2024

Про авторів:

Бухаленков Дмитро Олександрович,
магістрант НТУУ "КПІ імені
Ігоря Сікорського",
https://orcid.org/0009-0001-0224-8873
E-mail: 3a43mka@gmail.com

Заболотня Тетяна Миколаївна
Кандидат технічних наук
Кафедра програмного забезпечення
комп'ютерних систем
Національний технічний університет
України «Київський політехнічний
інститут імені Ігоря Сікорського»
Кількість статей в загальнодержавних
базах даних: 27
Кількість статей в міжнародних
базах даних: 2
H-index за Scopus: 2
ResearchGate: - ID Scopus: 6507406568
ResearcherID: J-2245-2017

Місце роботи авторів:

Національний технічний університет
України «Київський політехнічний
інститут імені Ігоря Сікорського»,
Берестейський проспект, 37, м. Київ,
Україна, індекс 03056
E-mail: zabolotnia@pzks.fpm.kpi.ua
ORCID: 0000-0001-8570-7571
Контактний тел.: +38-066-369-93-63

UDC 004.8 http://doi.org/10.15407/pp2024.01.023

V. R. Kobchenko, V.M. Shymkovysh, P.I. Kravets, A.O. Novatskyi, L.L. Shymkovysh, A.Yu. Doroshenko

AN INTELLIGENT CHATBOT FOR EVALUATING THE EMOTIONAL COLOURING OF A MESSAGE AND RESPONDING ACCORDINGLY

In this article, a recurrent neural network model, a database designed for neural network training, and a software tool for interacting with a bot have all been created. The architecture of the neural network model underwent optimization to enhance classification outcomes. Furthermore, work was conducted on enhancing the user interface. The developed application was tested, and the results were demonstrated. The resulting model demonstrated 85% accuracy in determining sentiments. The implemented application has got basic design which can be cutomized and some settings for chatbot. Further improvement of the model's classification quality can be achieved by collecting a larger and better organised dataset or by researching other RNN architectures.

Key words: recurrent neural networks, sentiment analysis, Python, tensorflow, keras, chatbot.

Introduction

Emotions in the broad sense are psychological and physiological reactions that arise in a person due to certain events, stimuli or thoughts. They are reflected through internal experiences and external expressions.

Usually, it is very easy to determine a person's emotions by observing their behavior, looking at their faces, or listening to their voices. However, the text itself does not have an emotional color. Instead, it can evoke emotions in the reader and reflect or express the emotional state of the author. This makes the task of assessing emotionality more difficult. And we usually don't need to know exact emotions, it is enough to know are they positive or negative.

This task is known as sentiment analysis [1,2] and is used in many areas of life. For example, processing reviews for products in a store can greatly simplify the work of marketers.

Besides marketing it is used in psychology, both for usual conversations and professional online help by psychologists.

And while it is not hard task for people, it can be still useful to automatize such analysis [3]. For marketers it will help, because they have to work with big number of reviews. Automatic detection of sentiments can facilitate the work of psychologists too. And this is possible to do with neural networks [4,5].

Artificial neural networks (ANNs) are computer models that imitate the functioning of biological neural networks [6,7]. Structurally they consist of interconnected artificial neurons that exchange signals in the form of numbers. There can be hundreds, thousands or even millions of such neurons. Each neuron receives input data, performs a certain function on this data and transmits the result to the next neurons in the network.

ANNs can be used in a wide range of applications, such as natural language processing, computer vision and speech recognition [7-17].

Recurrent Neural Networks (RNNs) are a special class of artificial neural networks [18,19]. Their feature is that they take into account the contextual dependence between elements of the sequence, that is, information about previous states to be taken into account in the current state.

In recurrent networks a directed sequence of elements is formed, presented in the form of a directed graph, and neurons are able to transmit it together with the processed data. This is achieved by using loops in each neuron.

1. Development of RNN

As already mentioned, the main feature of recurrent networks is the possibility of using pre-processed information for a better understanding of the existing information. This is a big step forward and it is actually effective. But does it always work? It all depends on how far down the sequence the context we need now is. If this "distance" is small, then everything really works well.

But it is quite clear that this is usually not the case and this "distance" can often be very large. At the same time, with its increase, the network at some point becomes unable to learn and combine the necessary information.

This problem is known as the gradient vanishing problem and is one of the main problems of recurrent neural networks.

Various modifications have been developed to overcome this problem. LSTM (Long Short-Term Memory) is one of those [20].

LSTM is an improvement of the recurrent neural network architecture that allows to study of long-term dependencies [21-23]. This allows such networks to cope with a large number of problems, thanks to which they have been widely used.

All recurrent neural networks have the form of a chain of repeated neural network modules. In standard RNNs, this chain has a simple structure, usually one layer with a certain activation function: hyperbolic tangent, sigmoid, etc.

LSTM also has a chain-like structure, but the module structure is different. Instead of one layer, there are four layers that interact with each other as shown in the figure 1.

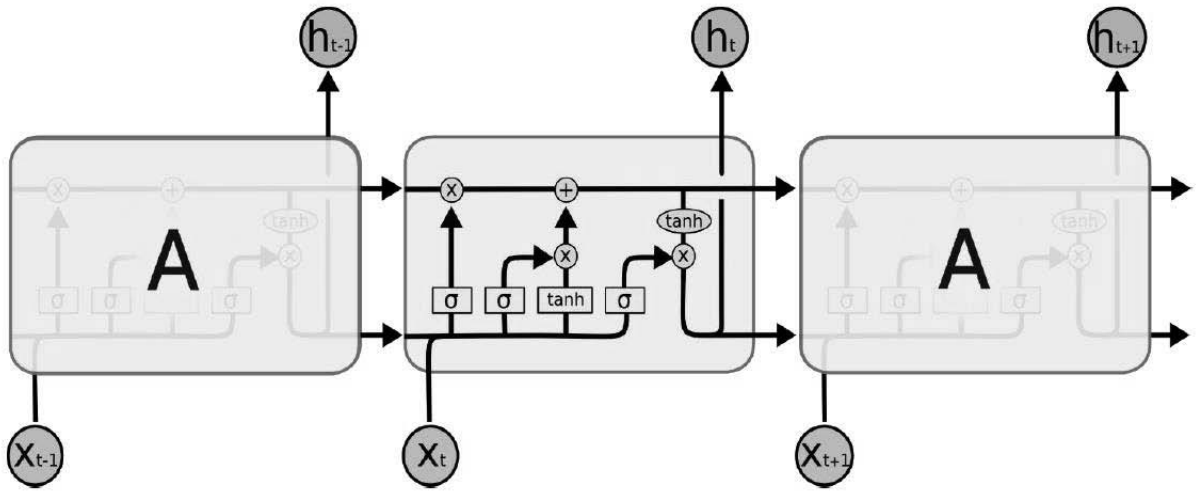

Figure 1. A recurrent LSTM module with four layers [24]

In the figure 1 each line carries a vector from the output of one node to the inputs of the others [24]. Pink circles represent element-by-element operations such as vector addition, and yellow rectangles represent trained neural network layers. Merged lines indicate concatenation, while split lines indicate copying of content, which is then sent to different locations.

The main idea behind LSTM is the cell state, the horizontal line that runs along the top of the chart.

The state of the cell is like a conveyor belt. It is simply laid along the entire chain with minor linear interactions. Information easily passes along the entire tape without changes.

But LSTMs have the ability to remove or add information to the state of a cell, which is controlled by structures called gates.

Filters are a way of controlling the transmission of information. They consist of a sigmoidal neural network layer and an element-by-element multiplication operation between the results obtained from the sigmoidal layer and the current state of the cell state.

The sigmoid layer outputs a number between zero and one, describing how much of each component should be transmitted. A

value of zero means "skip nothing" and a value of one means "skip everything".

LSTMs have three such filters to control the state of the cell.

The cell value is calculated in several steps. First, analyzing the information from the previous cell and deciding what can be removed from it. This decision is made by a sigmoidal layer called the "forgetting filter layer".

The next step is to decide what new information we will store in the cell state, and it consists of two parts. Firstly, the sigmoid layer, which called the "input filter layer", decides which values need to be updated. Secondly, the hyperbolic tangent (tanh) layer creates a vector of new values that can be added to the state.

After that updating the old state of cell C_{t-1} and obtaining a new C_t are performed. To do this, we multiply the old state by the value of f_t (the "forgetting filter layer"), thus forgetting what was decided to be forgotten in the first step. Then we add the product of it and C_t. These are the new potential values multiplied by the "importance" factor, which is how much we decide to update each state value.

Finally, we need to decide what we're going to output. This will be based on our cell state, but should also be filtered. First, a sigmoid layer is applied, which decides which parts of the cell's state we pass next. Then we pass the cell state through the tanh layer so that the values are in the range of -1 to 1 and multiply by the output values of the other sigmoid layer to output only what is needed.

This is the basic idea of LSTM. These have the ability to remember long-term dependencies, delete and add information to the cell's state, and control what will go to the output.

2. Development of a dataset for the training of model

This section describes the data collection process for training the LTSM model.

Data collection and preparation is an important stage in the process of developing and training a neural network [25], because we need to prepare the data on which it will learn.

The online platform Kaggle can help with this. Here you can find a large collection of various datasets for any purpose. There is also a proprietary development environment that can also be used to build neural networks. At the same time, the platform unites a community of people engaged in machine learning. Here you can ask anything on this topic or discuss something. Various competitions are also often held.

A dataset named IMDB Dataset of 50K Movie Reviews was found in Kaggle [26]. This is a collection of 50,000 movie reviews, each of which is marked with a sentiment (positive or negative) as shown in the figure 2.

The dataset is well balanced, so we have an almost equal number of positive and negative reviews.

Since the bot, which is being developed, must communicate in Ukrainian, the dataset was translated using the googletrans library for Python, which provides access to Google Translate from a python code file. Of course, the translation cannot be called exact, but

review	sentiment
A wonderful little production. The filming technique is very unassuming- very old-time- B...	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his par...	negative

Figure 2. Content of the dataset

translating 50 thousand reviews by hand would take too much time, so it was used.

The data also includes a list of stop words in the Ukrainian language. Fortunately, there are already ready-made lists, so one of these was used.

In addition, files were prepared with certain variants of appeals to the bot and probable reactions to such appeals. Scripts include greetings, farewells, thanks, and questions. And separately, of course, there is all the other text, which can be positive, negative or neutral, depending on which the bot will have a different reaction.

3. Description of application implementation

The developed application, with the help of libraries [27,28], consisted of three Google Colab[29] notebooks and three Python files.

The first notebook is used to translate the dataset into Ukrainian.

The second notebook is the main neural network, which is supposed to evaluate the emotional coloring of messages. First, the data is obtained from the dataset, the entire text is divided into training and test data, then the text is processed (removal of redundant characters using regular expressions and lemmatization using the pymorphy2 library).

The neural network is not able to work with the text, so the text is further tokenized. For this, all unique words that occur in the dataset are determined. A dictionary is formed from them, in which each word is given a unique token. This allows us to further represent the text as a list of tokens, where each token replaces a specific word. The dictionary is saved in a text file for later use. The data is also converted to the most suitable data format.

After that, the SentimentRNN neural network itself is created and trained on already processed data . Its structure is shown in Figure 3.

```
SentimentRNN(
  (embedding): Embedding(2001, 64)
  (lstm): LSTM(64, 256, num_layers=2, batch_first=True)
  (dropout): Dropout(p=0.3, inplace=False)
  (fc): Linear(in_features=256, out_features=1, bias=True)
  (sig): Sigmoid()
)
```

Figure 3. Structure of the SentimentRNN model

During training, a loss value is calculated for each epoch, and if it is less than the previous epochs, then the current state of the neural network is considered the best. The best state among all eras is saved to a file for later use.

The third notebook contains another neural network. Its task is to determine the intended scenarios in the user's messages, such as greetings, farewells, etc. In the data file for each scenario, certain patterns are defined, as well as responses to them.

Word processing is similar to the previous notebook, but word lemmatization is not used here.

The structure is much simpler as can be seen in Figure 4.

```
NeuralNet(
  (l1): Linear(in_features=30, out_features=8, bias=True)
  (l2): Linear(in_features=8, out_features=8, bias=True)
  (l3): Linear(in_features=8, out_features=4, bias=True)
  (relu): ReLU()
)
```

Figure 4. Structure of the NeuralNet model

Since it is not possible to create applications with a graphical interface in Google Colab, this part of the project was developed in Python IDLE, so we have three more files with the extension .py.

The first file contains the application interface, and the second and third are used to load the two models trained in Google Colab[29] and saved to files, and also contain functions for these models to predict the results.

4. Results of work and testing

If you look at the graphs of training and validation accuracies and losses, you can see that the results with the training data were constantly improving, while the best result on the validation data was achieved by the model at epoch 3, after which the losses only increased. This is indicative of overtraining and is clearly worth fixing, however 85% accuracy is a pretty good result, so the decision was made to leave this trained neural network as it is for epoch 3.

Then this is used in chatbot. An example of the conversation with it can be seen in figure 5.

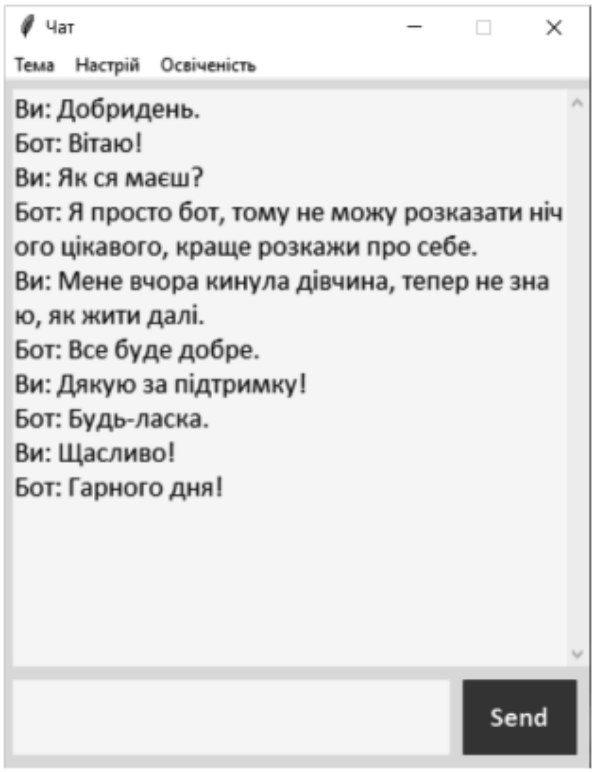

Figure 5. An example of a conversation with a developed chat bot

As you can see, the bot can recognize such common scenarios as greetings, thanks and goodbyes. When the user asks the bot about bot itself, it explains that it is an ordinary artificial intelligence and offers the user to tell what is on his mind.

The following is an example of what an average user could write to a bot. In this case, girl broke up with him, and we see that he is upset. And the bot, or rather the neural network behind it, is also able to recognize negative emotions, so it encourages the unhappy user in response.

Conclusions

In this article, a recurrent neural network with modification named long short-term memory was developed to analyze sentiments in messages.

A large dataset, 50 thousand comments, containing different reviews and their sentiments was collected and annotated to successfully train and validate the model. It was also translated into Ukrainian language.

The resulting model demonstrated accuracy 85% in determining sentiments. One more small model was developed to improve make chatbot recognise the intended scenarios in the user's messages.

And finally, chatbot application designed like an ordinary chat was created. There user can communicate with chatbot which uses pre-trained models to understand how user feels and responds reespectively. Also it has some customization such as changing design and bot behavior.

Further improvement of the model's classification quality can be achieved by collecting a larger and better organised dataset or by researching other RNN architectures.

References

1. Rodríguez-Ibánez, M., Casánez-Ventura, A., Castejón-Mateos, F., & Cuenca-Jiménez, P. M. (2023). A review on sentiment analysis from social media platforms. Expert Systems with Applications, 119862. https://doi.org/10.1016/j.eswa.2023.119862
2. Ravi, K., & Ravi, V. (2015). A survey on opinion mining and sentiment analysis: tasks, approaches and applications. Knowledge-based systems. Vol. 89, pp. 14-46. https://doi.org/10.1016/j.knosys.2015.06.015
3. Birjali, M., Kasri, M., & Beni-Hssane, A. (2021). A comprehensive survey on sentiment analysis: Approaches, challenges and trends. Knowledge-Based Systems. Vol. 226, 107134. https://doi.org/10.1016/j.knosys.2021.107134
4. Yadav, A., & Vishwakarma, D. K. (2020). Sentiment analysis using deep learning architectures: a review. Artificial Intelligence Review. Vol. 53(6), pp. 4335-4385. https://doi.org/10.1007/s10462-019-09794-5
5. Do, H. H., Prasad, P. W., Maag, A., & Alsadoon, A. (2019). Deep learning for aspect-based sentiment analysis: a comparative review. Expert systems with applications. Vol. 118, pp. 272-299. https://doi.org/10.1016/j.eswa.2018.10.003
6. Sarker, I. H. (2021). Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. SN Computer Science. Vol. 2(6), 420. https://doi.org/10.1007/s42979-021-00815-1
7. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), pp. 436-444. https://doi.org/10.1038/nature14539
8. Shymkovych V., Telenyk S., Kravets P. (2021) Hardware implementation of radial-

basis neural networks with Gaussian activation functions on FPGA. Neural Computing and Applications. vol. 33, no.15, pp. 9467-9479. https://doi.org/10.1007/s00521-021-05706-3

9. Harumy, T. F., Zarlis, M., Effendi, S., & Lidya, M. S. (2021, August). Prediction using a neural network algorithm approach (a review). 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM), Pekan, Malaysia, IEEE, pp. 325-330. https://doi.org/10.1109/ICSECS52883.2021.00066
10. Shymkovych, Volodymyr, Anatoliy Doroshenko, Tural Mamedov, and Olena Yatsenko (2022) Automated Design of an Artificial Neuron for Field-Programmable Gate Arrays Based on an Algebra-Algorithmic Approach. International Scientific Technical Journal "Problems of Control and Informatics" vol. 67, no. 5, pp. 61-72. https://doi.org/10.34229/2786-6505-2022-5-6
11. Perera, N. N., & Ganegoda, G. U. (2023). A Comprehensive Review on Speech Synthesis Using Neural-Network Based Approaches. 2023 3rd International Conference on Advanced Research in Computing (ICARC), Belihuloya, Sri Lanka, IEEE, pp. 214-219 https://doi.org/10.1109/ICARC57651.2023.10145741
12. Bezliudnyi Y., Shymkovysh V., Doroshenko A.(2021) Convolutional neural network model and software for classification of typical pests. Prombles in programming. Vol.4, pp. 95-102. https://doi.org/10.15407/pp2021.04.095
13. Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. Multimedia tools and applications. Vol. 82(3), pp. 3713-3744. https://doi.org/10.1007/s11042-022-13428-4
14. Kravets P., Nevolko P., Shymkovych V., Shymkovych L. (2020) Synthesis of High-Speed Neuro-Fuzzy-Controllers Based on FPGA. 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT). pp. 291-295. https://doi.org/10.1109/ATIT50783.2020.9349299
15. Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Machine Learning with Applications. Vol. 6, 100134. https://doi.org/10.1016/j.mlwa.2021.100134
16. Kravets, P., Novatskyi, A., Shymkovych, V., Rudakova, A., Lebedenko, Y., Rudakova, H. Neural Network Model for Laboratory Stand Control System Controller with Parallel Mechanisms. Lecture Notes on Data Engineering and Communications Technologies. Springer, Cham. 2023. Vol 181. pp. 47-58 https://doi.org/10.1007/978-3-031-36118-0_5
17. Y.S. Hryhorenko, V.M. Shymkovysh, P.I. Kravets, A.O. Novatskyi, L.L. Shymkovysh, A.Yu. Doroshenko. A convolutional neural network model and software for the classification of the presence of a medical mask on the human face. Problems in programming. 2023. Vol. 2. pp. 59-66. https://doi.org/10.15407/pp2023.02.059
18. Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. Neural computation. Vol. 31(7), pp. 1235-1270. https://doi.org/10.1162/neco_a_01199
19. Bezliudnyi Y., Shymkovych V., Kravets P., Novatsky A., Shymkovych L. Pro-russian propaganda recognition and analytics system based on text classification model and statistical data processing methods. Адаптивні системи автоматичного управління: міжвідомчий науково-технічний збірник. 2023. № 1 (42), с. 15-31. https://doi.org/10.20535/1560-8956.42.2023.278923
20. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation. Vol. 9(8), pp. 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
21. Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. Artificial Intelligence Review. Vol. 53, pp. 5929-5955 https://doi.org/10.1007/s10462-020-09838-1
22. Kader, N.I.A., Yusof, U.K., Khalid, M.N.A., Husain, N.R.N. (2023). A Review of Long Short-Term Memory Approach for Time Series Analysis and Forecasting. Lecture Notes in Networks and Systems. Vol 573. Springer, Cham. pp. 12-21 https://doi.org/10.1007/978-3-031-20429-6_2
23. Long, F., Zhou, K., & Ou, W. (2019). Sentiment analysis of text based on bidirectional LSTM with multi-head attention. IEEE Access,7, pp. 141960-141969. https://doi.org/10.1109/ACCESS.2019.2942614

24. Olah, C. (2015). Understanding LSTM networks. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
25. R. Yu, S. Liu, X. Wang (2024) Dataset Distillation: A Comprehensive Review. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 46, no. 1, pp. 150-170. https://doi.org/10.1109/TPAMI.2023.3323376
26. IMDB Dataset of 50K Movie Reviews URL: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
27. TensorFlow. (n.d.). TensorFlow: An end-to-end open source machine learning platform. Retrieved from https://www.tensorflow.org/
28. Keras. (n.d.). Keras: The Python deep learning API. Retrieved from https://keras.io/
29. Google Colab https://colab.research.google.com/

Одержано: 16.01.2024

Про авторів:

Кобченко Владислав Русланович,
студент магістр НТУ України
«КПІ імені Ігоря Сікорського»

Шимкович Володимир Миколайович,
кандидат технічних наук, доцент НТУ України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – понад 30.
Кількість наукових публікацій
в зарубіжних виданнях – понад 10.
Індекс Хірша – 4. https://orcid.org/0000-0003-4014-2786

Новацький Анатолій Олександрович,
кандидат технічних наук,
доцент НТУ України
«КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – понад 30

Кравець Петро Іванович,
кандидат технічних наук, доцент НТУ України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – понад 40.
Кількість наукових публікацій
в зарубіжних виданнях – понад 10.
Індекс Хірша – 4. https://orcid.org/0000-0003-4632-9832

Шимкович Любов Леонідівна,
асистент кафедри інформаційних систем та технологій НТУ України
«КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – 4.
Кількість наукових публікацій
в зарубіжних виданнях – 1.
https://orcid.org/0000-0002-1291-0373

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, професор НТУ України
«КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – понад 200.
Кількість наукових публікацій
в зарубіжних виданнях – понад 90.
Індекс Хірша – 7. http://orcid.org/0000-0002-8435-1451

Місце роботи авторів:

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», проспект Перемоги 37 та Інститут програмних систем НАН України, 03187, м. Київ-187, проспект Академіка Глушкова, 40.
E-mail:
vladik020402@gmail.com,
v.shymkovych@kpi.ua,
a.novatskyi@kpi.ua,
peter_kravets@yahoo.com,
L.shymkovych@gmail.com,
doroshenkoanatoliy2@gmail.com

УДК 517.958:57 +519.711.3 + 612.51.001 http://doi.org/10.15407/pp2024.01.030

R.D. Grygoryan

PROBLEMS ASSOCIATED WITH CREATING SPECIAL SOFTWARE FOR SIMULATING OF HUMAN PHYSIOLOGICAL RESPONSES TO DYNAMIC $\pm G_z$ ACCELERATIONS

Under extreme accelerations, human physiological mechanisms cannot provide adequate circulation. Special methods and devices protecting pilot's brain and eye functionality have been proposed but their efficiency is individual and depends on pilot's skills. Currently, the lonely technology to safely acquire and test the necessary skills is based on use of special centrifuges. However, lack of adequate data about physiological and biomechanical events are two main causes worsening the training results. Special computer simulators, capable to model and visualize the main mechanical and physiological effects occurring under dynamic accelerations, could increase the effectiveness of future pilot's training process. This publication aims to define fundamental problems concerned with creating the required software. There exist two main groups of problems. The first group is concerned with the necessity to create basic mathematical models quantitatively describing both the physiological events and effects induced by protective maneuvers. Here special logical procedures, individualizing the basic physiological models, have to be proposed. The second group of problems is predominantly technical and associated with the necessity of special user interface (SUI) development. SUI must be subdivided into two functional sections – one for preparing a single computer experiment (simulation), and another – for analyzing the results of simulation. An experiment preparation includes the following events: i) a preliminary tuning of models according to biometrical data; ii) a setting of acceleration profile; iii) a choosing of protective algorithms and tools (or without protections); iv) a choosing of forms for results storage. Graphs presenting the dynamics of input and output variables are the main forms while the table forms are also included. The user (trainer or trainee) will be able to retrieve from the memory graphs of previous simulations to compare the effectiveness of additional protective elements. The software must be autonomic for the Windows platform.

Keywords: human extreme physiology, quantitative models, simulator, training, information technology.

Introduction

Maneuvers on modern fighter aircraft are associated with rapid altering and often highly sustained extreme accelerations [1-3]. Both physiological [4-9] and biotechnical [10-12] problems that arose in parallel with an increase in military aircraft's maneuverability have been properly investigated [4-18]. Human physiology evolutionarily adapted to the one g Earth environment, cannot provide adequate functioning of the brain and eyes of a sitting person. Two of these organs, very sensitive to oxygen and glucose supply, suffer in parallel with the decreasing of their input blood pressure. Under accelerations, the hydrostatic pressure proportional to the acceleration value expands the vascular wall, accumulating greater blood volume. The altered pressure gradients do not provide the necessary circulation at the cardiovascular scale. Accelerations also alter the ventilation-perfusion ratio in lungs [13,14].

Most critical are positive (+Gz) accelerations acting in the direction of head-legs, or negative (-Gz) accelerations acting in the opposite direction [4-6]. In terminal zones (brain, eyes), the lowered circulation causes oxygen lack and worsens the pilot's vision and consciousness [9,12]. Under -Gz, the elevated local blood pressure in the eyes and brain causes rupture of microscopic vessels and hemorrhages. Both the value of Gz and the gradient of acceleration change play an essential role in these events.

Under relatively slow (0.1-0.4 g/sec) linearly increasing +Gz accelerations, a mean healthy person not using artificial protections is operable for approximately +4Gz accelerations [11]. Further elevation of the G-load causes the G-lock phenomenon usually disappearing after a break [2,6,8].

Modern fighter aircrafts (like F16, F35, and others) can provide acceleration gra-

dients exceeding 2 g/sec. This requires special protection algorithms and devices. Currently, typical protection algorithms include the use of special pneumatic or water-augmented anti-G suits, muscle stress, as well as breathing with a positive pressure air [1,11,17]. The adaptive protection algorithms combining multiple methods depending on the dynamics of accelerations are the most effective. So, a technology helping to optimally combine protective methods and tools is encouraged.

Traditionally, empiric research on centrifuges is the main way for inventing more effective protections [1,5,6-8]. As was demonstrated in [18-20], mathematical models realized as special software provided by additional ways to maximize the individual resistance of a pilot to the negative effects of accelerations. The experience in this special area is a basis for creating an advanced version of such software.

This article defines the main requirements for future software and ways for its creation.

Main functional blocks of the future software

1. The main blocks of models

The main mathematical models conventionally divided into two groups are shown in Fig.1.

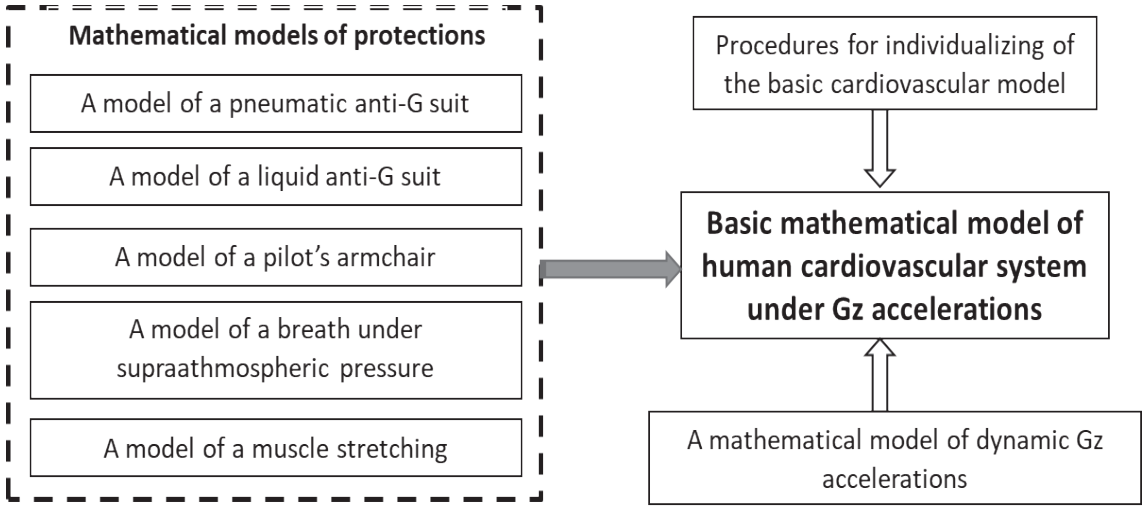

Figure1. Mathematical models and procedures to be used in the future software.

Fig.1 indicates that there should be created two blocks of quantitative models: models of physiological mechanisms; and models representing environmental physical factors modulating local, regional, or total hemodynamics.

Models of physiological mechanisms should quantitatively describe the dynamics of blood pressures, volumes, and flows of a sitting person under altering blood hydrostatic pressure. Important is that the model must describe the main neural-hormonal influences (modulations) of characteristics, involved in descriptions of both the cardiac pump function and vascular tonus.

A correct understanding of the student (future pilot) of the essence of physiological processes during flight overloads can play an essential role in the acquisition of professional skills. Usually, empirical technologies for training pilots to counteract the undesirable effects of g-forces focus on two main circumstances caused by extreme accelerations: 1) narrowing of the field of peripheral vision or loss of vision; 2) loss of consciousness as an extreme manifestation (G-lock). These phenomena, caused by a deterioration of the eyes and brain oxygen supply, are only manifestations of more extensive changes in human hemodynamics. However, empirics provide very scant information about these hemodynamic processes. The matter is that standard measurements are limited to monitoring the dynamics of heart rate (HR) and blood pressure. The shift in HR can be provided by multiple mechanisms. Several of them are

known as physiological regulators reacting to a drop in pressure in the reflexogenic zones of the arterial tree. Other regulators can be largely activated by the mechanical stretching of body structures. Therefore, the proper model must describe the effects of both mechanisms. Finally, the specific dynamics of blood pressure are associated with movements and redistribution of significant volumes of blood under the influence of increased hydrostatic pressure.

The quantitative mathematical model of human hemodynamics is the single method that can illustrate the cause-and-effect relationships of developing dynamic events. It should be emphasized that the maximum additive effect of protective agents can only occur when each additional protective agent is activated at the right time. It can be detected using the mathematical model we create.

To increase the visibility of the protective effect, a special simulation mode will be provided when the physiological regulators are turned off. This simulation will reveal what could happen to hemodynamics if the body is late to respond to the mechanical movements of blood in the human body.

In Fig. 1, the second group of models collects models quantitatively describing the influences of external mechanical dynamic forces on local, regional, or total hemodynamics. Namely, these models describe hemodynamic modulations of each protective tool and algorithm.

In the right part of the Fig. 1, two additional options are indicated. The bottom-located rectangle accentuates the fact that a special model will be proposed for the simulation of acceleration profiles. At last, the upper rectangle indicates that special logic and mathematic approximations must be proposed to individualize the physiology model using anthropometrics and the sex of the person to be tested (simulated).

At last, the software must provide a computer experiment (simulation) and recording of its results as a special experiment protocol (SEP). Personalized records of SEPs can be accumulated in special files for their reproducing and deep analysis. The latter must provide an option for comparing of chosen variables for at least two experiments. These basic requirements mainly determine the software architecture.

The future software is intended to be autonomous Windows oriented. Exe module called “Accel.exe”. The success of software depends on two main factors: 1) how much it is needed; and 2) how practical is its user interface (UI). In our particular case, UI must be oriented both to student pilots and their trainers-instructors. Therefore, icons intuitively appointing procedures needed to be activated for the program’s preparation before executing is desirable. In addition, special icons intuitively appointing procedures for visualizing and analysis of simulation results are encouraged. Fig.2 below indicates the main procedures necessary for preparing and executing a single computer experiment (simulation), and that have to be provided by a UI.

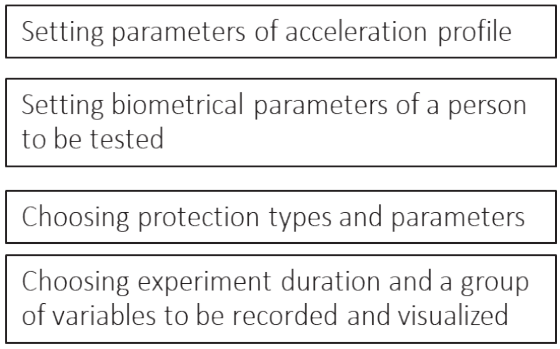

Fig.2. Procedures that are necessary for preparing and executing a single computer experiment (simulation), and that have to be provided by a special user interface (UI).

2. Main functional blocks of UI

By downloading the future autonomous “Accel.exe”, the user will have a core (basic version) of the quantitative model that has to be specially tuned for providing a single computer experiment (simulation). The tuning operations, provided by the UI, are intended to transform the basic version of complex models to a person-oriented model possessing a set of constants characterizing both physiological and environmental parameters.

Personal sensitivity of nervous regulators can vary in a wide range. There is no strain recommendation for associating the sensitivity coefficients with anthropometric data. At the same time, it is known that some tests (in particular, the postural test) can help

us to approximately individualize the heart rate's neurogenic sensitivity. During the project execution, several ideas have to be algorithmically realized and tested.

Mathematical models

The principle is to differentiate two blocks of mathematical models. The first block describes human hemodynamics under varying hydrostatic pressures and extravascular pressures. The model does take into account the main effects of physiological control mechanisms that normally provide acute cardiovascular responses to dynamic accelerations.

1. The background of the physiological models

The human cardiovascular system (CVS) must be presented in the model as a structure combined with two subunits. The first subunit represents the vascular bed as a net of arterial and venous compartments each localized at different distances from the foot level. Every vascular compartment will have its fixed initial parameters (rigidity $D_i(0)$, unstressed volume $U_i(0)$, and resistance $r_i(0)$). Current (dynamic) values of these variables will be calculated taking into account increments provided by nervous-endocrine regulators:

$$U_i(t) = U_i(0) \pm \sum_j \Delta U_j, \qquad (1)$$

$$D_i(t) = D_i(0) \pm \sum_j \Delta D_j, \qquad (2)$$

$$r_i(t) = r_i(0) \cdot \mathrm{Z}(D_i(t), U_i(t)). \qquad (3)$$

The principal is to model the influences of gravitational and extravascular mechanical forces on local output blood flows $q_i(t)$:

$$q_i(t) = [(P_i(t) + P_i^G(t) + P_i^E(t)) - (P_{i+1}(t) + P_{i+1}^G(t) + P_{i+1}^E(t))] / r_i(t), \qquad (4)$$

where the gravitational component $P_i^G(t)$ is calculated using the distance of the vascular compartment from the foot h_i, the current value of acceleration $G_z(t)$, and the angle $\alpha_i(t)$ between acceleration vector and body vascular compartment.

$$P_i^G(t) = \gamma \cdot h_i \cdot G_z(t) \cdot \sin \alpha_i(t). \qquad (4.1)$$

As observation intervals under simulating of acceleration events are limited by minutes, the total blood volume $V = const.$ Shifts of compartmental ($V_i(t)$) and regional blood volumes appear due to alterations of compartmental input-output flows:

$$\frac{dV_i}{dt} = q_{i-1}(t) - q_i(t). \qquad (5)$$

Arterial baroreceptor reflexes are considered to be the main physiological mechanisms counteracting to lowering of perfusion pressure in the brain [67]. In a rigid cranium, the normal extravascular pressure is slightly subatmospheric. +Gz accelerations, especially in non-collapsible venous sinuses, aggravate the negative pressure. These biomechanical factors, lowering the venous return from a cranial basin to the heart, play a specific counteracting role against the gravitational decrease of brain circulation. Besides, within 50 mm Hg alterations of brain perfusion pressure, autonomic nervous mechanisms provide a practical constancy of the summary brain flow. Additional protective effects are provided by a reflector mechanism activated due to the lowering of intravascular pressure in the circle of Willis. This effect was first shown in [78].

Practically always real military fighter missions are accompanied by mental stress and delivery of catecholamines that additionally mobilize CVS. Empiric observations have also shown a phenomenon of heart rate extreme increase despite essentially elevated blood pressure in the aortic arch. This phenomenon can be explained and modeled in the assumption that under high levels of acceleration, the mechanical stress of muscles originates high proprioception capable of compensating the depressor influence of the aortic arch baroreflex.

2. The background of models describing the physical environment

Fighter pilots, functioning in a highly dynamic environment, are provided by artificial specifically acting biomechanical protec-

tive tools and methods. Standardly, they include: i) a pilot's chair with a declined to horizon under angle A supine; ii) anti-G suit; iii) a helmet provided by a device for breathing under positive pressure. To this list can be added special technics for tensing the muscles of the legs and abdominals.

So, to simulate the protective effects of the artificial tools, additional equations describing the transformation of external biomechanical forces into the body's regional vasculature are required. Simulation algorithms must provide applications of a chosen acceleration profile for every combination of protections. To demonstrate protection effects, as well as to compare the effects of every single method, it desirable is to have two special model versions. The first one illustrates hemodynamics under accelerations with switched-of physiological regulators. The second version does simulate hemodynamic responses of a person with the normal functioning of the physiological controllers of CVS. There are three specific ways to counteract extreme increases in vascular volume: 1) increasing the vascular rigidity; 2) decreasing the non-stressed volume; 3) elevating the extravascular pressure.

The first two ways are incorporated into physiological reflector regulators. The third opportunity can be used without or with the application of artificial devices. In the first case, the person being under acceleration forces can consciously increase the tension in the muscles of the legs and abdominals. When additional efforts to exhale with a closed airway have been provided, the protective effect is higher.

Constructively, anti-G suits can be with pneumatic or water-filled chambers. The standard anti-G suit is sectional (for body sections of the abdomen, thighs, and shins). Versions of suits containing special sections for creating certain supra-atmospheric pressure at the chest are also proposed [6]. In general, these suits do resist the accumulation of local blood volumes. So, in the background of +Gz, a pumping of pressured air into the anti-G suit lowers the sectional blood volume. The matter is how to model these protective effects.

In equation (4) above, there is a variable $P_i^E(t)$ which presents local extravascular pressure. Certainly, leg muscles, abdomen, or thoracic cavities will have their transfer coefficients for transmitting the applied external pressure ($P_E(t)$) to the depth of blood vessels. In (4), summary extravascular pressure should be calculated as:

$$P_i^E(t) = k_i \cdot P_E(t) \ . \qquad (4.2)$$

3. Models describing input loads

The model to be used in the future software will calculate human hemodynamic responses to two different types of input dynamic loads. During simulations of the pilot who is not using protections, the dynamics of accelerations are the lonely input loads initiating alterations of the normal physiology of circulation and causing reflector responses. Protection activation should be considered as the second type of input load. Real flight maneuvers, especially combat maneuvers, occur under combined input loads. To estimate the separate effects of each input load, it is necessary to design algorithms providing arbitrary combinations of both types of input loads. Below, supposing the value of A to be constant, the first and the second types of input loads are consequentially considered.

The initial hemodynamic alterations depend on both acceleration gradients and amplitude. So, to correctly simulate these biomechanical events, our software must include a model, describing accelerations dynamics. It is planned to provide the following acceleration profiles: a) a linear with a given gradient a and of τ duration; b) a linear with a further transforming to a constant (plateau) of τ duration; c) a trapezoidal with given parameters; d) a special profile for imitating the known "push-pull" effect; e) arbitrary constructing by a user profile. Formalisms for modeling these profiles are described below.

The first profile is presented as:

$$a(t) = \begin{cases} 0, & t \le t_s \\ A_g(t - t_s), & t > t_s \end{cases}, \qquad (6)$$

where t_s - is start time of the increasing of acceleration with a gradient of A_g .

The second profile is presented as:

$$a(t)=\begin{cases} 0, & t \le t_s \\ A_g(t-t_s), & t < t_s \le t_p \\ a_m, & t_p < t \le t_p + T \end{cases}, \qquad (7)$$

where t_p - is the time of reaching a plateau level with acceleration value of a_m.

The trapezoidal profile formalism looks as:

$$a(t)=\begin{cases} 0, & t \le t_s; a_m \le 0 \\ a_g(t-t_s), & t < t_s \le t_p \\ a_m, & t_p < t \le t_p + T \\ a_m - d_g(t-t_p-T), & t > t_p + T \end{cases}, \qquad (8)$$

where d_g - is the deceleration gradient, T - is the plateau's duration.

Formalisms describing the second type of the input loads look as:

$$p(t)=\begin{cases} 0, & a(t) \le 0 \\ \Delta_j \cdot a(t), & a_1 < a(t) \le a_m \\ \Delta_j \cdot a_m - d_g(t), & 0 < a(t) < a_m \end{cases}, \qquad (9)$$

where Δ_j can be altered step-like.

Equations (1)-(9) were involved in this publication to help readers to understand our approach to the modeling. Much detailed information one can find in [18-21]. The final version of the mathematical model describing the formal basis of the future software and simulation results will be published separately.

Requirements to input and output data organization

As already mentioned above, the "Accel.exe" is planned to be autonomic software that provides an option for personalization. To satisfy these requirements, special algorithms are needed. They should provide initial data for characteristics of every vascular compartment, as well as for physiological regulators. In fact, this is the basic version of physiological models (BVPM). All additional procedures for BVPM's personalizing using the input data must be algorithmically organized.

1. The input data

BVPM should be tuned for the hemodynamics of a healthy man with mean anthropometric parameters of mas (M) and, height (H). Special formulas used M to calculate the total blood volume (V) must be provided. Then, utilizing incorporated special massive of coefficients must be distributed in four cardiac chambers and 33 vascular (arterial and venous) compartments. Coefficients will characterize human horizontal position sectional or regional blood volumes. The next procedure does provide automatic recalculating of these initial blood volumes to new ones characteristic for the relaxed human in a standard sitting position on a pilot armchair. Equations for these recalculations must take into account the initial parameter V. Individual anatomical peculiarities (if they are) concerning lengths of neck, tights, and shins have to be taken into account.

2. The output data

There are planned two types of output data: one by default, and another – in special cases. The default data concerning acceleration profile, arterial pressures in three body zones (brain, eyes, aortic arch), and HR should have been presented in graph forms. This form is recommended to both student-pilots and instructors. Additional data including inside information about parameters of models is best to collect in table forms. Experts developing advanced protections can be the users of these data. Namely, this class of users can prepare and simulate novel combat maneuvers avoiding risks for testers.

Conclusion

Combat maneuvers of modern fighter aircraft originate extreme accelerations negatively influencing on pilot's physiology and operability. Empirical investigations were the only way to develop and test protective methods and tools providing fighter pilots functionality under combat maneuvers. The main tools used for acquiring student pilot initial skills necessary to resist the negative effects of dynamic extreme accelerations were centri-

fuges. The skilling process of student pilots of modern fighter aircraft is not duly formalized yet. Potentially, special computer simulators providing additional data concerning characteristics of both human physiology and protective methods under dynamic accelerations could improve the skilling process and its efficiency. Certain scientific-practical problems associated with creating the needed simulator have been considered in the paper. To create special software, quantitative mathematical models must be previously created. They represent both the human cardiovascular physiology and protective technologies under exposure to sustained and extreme $\pm G_z$ accelerations.

References

1. Burton, R.R.; Whinnery, J.E. Biodynamics: Sustained acceleration. In Fundamentals of Aerospace Medicine, 3rd ed.; DeHart, R.L., Davis, J.R., Eds.; Lippincott Williams & Wilkins: Philadelphia, PA, USA, 2002; pp. 122–153.
2. Slungaard E., McLeod J., Green, N.D.C., Kiran A., Newham D.J., Harridge S.D.R. Incidence of g-induced loss of consciousness and almost loss of consciousness in the Royal Air Force. Aerosp. Med. Hum. Perform. 2017, 88, 550–555.
3. Newman, D.G. The cardiovascular system at high Gz. In High G Flight: Physiological Effects and Countermeasures, 1st ed.; Newman, D.G., Ed.; Ashgate: Farnham, UK, 2015; pp. 57–72.
4. Park, M.; Yoo, S.; Seol, H.; Kim, C.; Hong, Y. Unpredictability of fighter pilots' G duration by anthropometric and physiological characteristics. Aerosp. Med. Hum. Perform. 2015, 86, 307–401.
5. Yun, C.; Oh, S.; Shin, Y.H. AGSM proficiency and depression are associated with success of high-G training in trainee pilots. Aerosp. Med. Hum. Perform. 2019, 90, 613–617.
6. Polock,R.D.,Hodkinson,P.D.,&Smith,T.G.Oh G:Thex,yandzofhumanphysiologicalresponses toacceleration. Experimental Physiology, 2021,106,2367–2384. https://doi.org/10.1113/EP089712
7. Albery, W. B. Acceleration in other axes affects +Gz tolerance: Dynamic centrifuge simulation of agile flight. Aviation, Space, and Environmental Medicine, 2004,75(1), 1–6.
8. Burton, R., & Whinnery, J. Operational G-induced loss of consciousness: Something old; something new. Aviation, Space, and Environmental Medicine, 1985,56(8), 812–817.
9. Cao, X.-S., Wang, Y.-C., Xu, L., Yang, C.-B., Wang, B., Geng, J., Gao, Y., Wu, Y. H., Wang, X. Y., Zhang, S., & Sun, X.-Q. Visual symptoms and G-induced loss of consciousness in 594 Chinese Air Force aircrew— A questionnaire survey. Military Medicine, 2012, 177(2), 163–168. https://doi.org/10.7205/milmed-d-11-00003.
10. Chung, K. Y. Cardiac arrhythmias in F-16 pilots during aerial combat maneuvers (ACMS): A descriptive study focused on G-level acceleration. Aviation, Space, and Environmental Medicine, 2001,72(6), 534– 538.
11. Eiken, O., Bergsten, E., & Grönkvist, M. G-Protection mechanisms afforded by the anti-G suit abdominal bladder with and without pressure breathing. Aviation, Space, and Environmental Medicine, 2011. 82(10), 972–977. https://doi.org/10.3357/ASEM.3058.2011
12. Eiken, O., Keramidas, M. E., Taylor, N. A. S., Grönkvist, M., & Gronkvist, M. Intraocular pressure and cerebral oxygenation during prolonged headward acceleration. European Journal of Applied Physiology, 2017,117(1), 61–72. https://doi.org/10.1007/s00421-016-3499-3
13. Grönkvist, M., Bergsten, E., & Eiken, O. Lung mechanics and transpulmonary pressures during unassisted pressure breathing at high Gz loads. Aviation, Space, and Environmental Medicine, 2008,79(11), 1041–1046. https://doi.org/10.3357/ASEM.2371.2008.
14. Henderson, A. C., Sá, R. C., Theilmann, R. J., Buxton, R. B., Prisk, G. K., & Hopkins, S. R. The gravitational distribution of ventilationperfusion ratio is more uniform in prone than supine posture in the normal human lung. Journal of Applied Physiology, 2013, 115(3), 313–324. https: //doi.org/10.1152/JAPPLPHYSIOL.01531.2012
15. MacDougall, J. D., McKelvie, R. S., Moroz, D. E., & Buick, F. The effects of variations in the anti-G straining maneuver on blood pressure at +Gz acceleration. Aviation, Space, and Environmental Medicine, 1993,64(2), 126–131.
16. Sundblad, P., Kölegård, R., Migeotte, P. F., Delière, Q., & Eiken, O. The arterial baroreflex and inherent G tolerance.EuropeanJournal of Applied Physiology, 2016,116(6), 1149–1157. https://doi.org/10.1007/s00421-016-3375-1

17. Whiny,J.E.,&Forster,E.The+Gz-induced loss of consciousness curve. Extreme Physiology and Medicine, 2013,2(1),19. https://doi.org/10.1186/2046-7648-2-19
18. Grygoryan, R.D., Kochetenko, E.M., Informational technology for modeling of fighters medical testing procedures by centrifuge accelerations. Selection &Training Advances in Aviation: AGARD Conference Proceedings 588; Prague, 1996. May 25-31, PP3,1-12.
19. Grygoryan, R.D., 2002. High sustained G-tolerance model development.STCU#P-078 EOARD# 01-8001 Agreement: Final Report. 66p.
20. Grygoryan R.D. Problem-oriented computer simulators for solving of theoretical and applied tasks of human physiology. Problems of programming. 2017, №3, 102-111.
21. Grygoryan R.D. Modeling of mechanisms providing the overall control of human circulation. Advances in Human Physiology Research, 2022,4, 5 – 21, https://doi.org/10.30564/ahpr.v4i1.4763.

Received: 05.02.2024

.

About authors:

Grygoryan Rafik,
Department chief, PhD, D-r in biology
Publications number in Ukraine journals -124
Publications number in English journals -39.
Hirsch index – 11
http://orcid.org/0000-0001-8762-733X

Place of work:

Institute of software systems of
Ukraine National Academy of Sciences
03187, Kyïv,
Acad. Glushkov avenue, 40,
E-mail:rgrygoryan@gmail.com

UDK 621.09.04 http://doi.org/10.15407/pp2024.01.038

S. V. Strutynskyi, V. A. Yalanetskyi

PROGRAMMING OF ONE-DIMENSIONAL AND TWO-DIMENSIONAL TOKENS FOR TOKENIZATION OF LAND PLOTS

The use of blockchain tools that allows spliting virtual objects into parts is considered. Examples of practical use of the developed algorithms are presented. The concept of one-dimensional and two-dimensional tokens representing one-dimensional and flat objects is proposed. Algorithms for the implementation of one-dimensional tokens are developed, and the peculiarities of their practical application are considered. A designed smart contract allows to conduct a basic list of operations with one-dimensional tokens. Algorithms, providing implementation of two-dimensional tokens, are proposed. Peculiarities of presenting territories of virtual worlds and land plots are suggested. A comparative analysis of the use of NFT and two-dimensional tokens for presenting the Earth surface areas is performed. Methods that ensure ownership of tokens at different levels are proposed.
Keywords: Blockchain, EVM, smart contracts, NFT, fractional tokens.

Introduction

Blockchain technologies proved their effectiveness within creating distributed databases. They found many applications, including cryptocurrencies; tokenized assets presenting fiat currencies, shares, property and other financial instruments; non-fungible tokens representing digital objects; decentralized domain name databases [1].

Some of the tools and solutions based on blockchain technologies have been successfully implemented. Algorithms and software tools that have proven their suitability for practical use and resistance to attacks have been developed. However, current tools have limited capabilities preventing the creation of specialized blockchain-based solutions.

One of the long-term areas of using blockchain technologies is the implementation of database of land plots. Decentralized applications will allow transactions with real estate, land plots and virtual territories. The implementation of solutions, providing an opportunity to perform specialized transactions, requires the deployment of new blockchain tools. An infrastructure that is optimized for working with land plots and virtual territories are to be created.

The tools that are necessary for real estate transactions are at the initial stage of development. There are algorithms and simplified solutions that, if customized, could be widely used.

Analysis of recent research and publications

Cryptocurrencies, tokens and other virtual assets are characterized by different tokennomics. Virtual assets could be generated once within startup or periodically after certain time intervals. NFTs might have more complex emission mechanisms. Most cryptocurrencies and tokens have a variable volume of emission. If the number of coins in circulation fluctuates significantly, this leads to some difficulties, e.g. when determining the current asset capitalization.

Coins in the Bitcoin network are issued to pay miners who generate blocks [2]. The exponential decay of the payment for the creation of the block led to the fact that the volume of funds in circulation ceased to change significantly. The additional emission is insignificant compared to the number of coins in circulation.

Some cryptocurrencies are characterized by constant emission. Payments to miners or validators are conducted from funds paid by customers for the performance of transactions. Such an algorithm reduces inflationary trends significantly.

There are also cryptocurrencies whose emission volume is decreasing. The process is implemented by burning virtual assets. The source of funds could be commission fees, a part

of which is withdrawn from circulation. An example of the implementation of such an algorithm is the Ethereum blockchain platform [3]. After the activation of the London hard fork, which includes EIP-1559 [4], a part of the commissions is withdrawn from circulation. The established algorithm is quite flexible, since the volume of ETH emission could decrease or increase according to external factors.

The proposal of stablecoins and tokenized assets is determined by the management company. Most often, their emission of tokens depends on the market demand for them.

Non-fungible tokens are put into circulation through other algorithms. The emission of tokens is implemented via smart contract, which allows you to link NFTs with a digital object. Most often, the token contains a file hash. In this case, the file is not stored on the blockchain, but the hash allows you to identify the data-set and link it to the NFT. In some cases, additional algorithms can be used when performing operations with tokens. In some cases, the author of the NFT can receive royalties from the sale of the token [5].

Financial instruments based on blockchain are characterized by different volume of emission. Most often, the emission of an asset is measured in millions, billions or quadrillion units. A large volume of emission provides customers with certain conveniences when conducting small transactions.

Cryptocurrencies and ERC-20 tokens [6] issued in EVM-compatible networks are characterized by a large volume of emission, for example10^{26}. This is caused by current limitations of Solidity programming language when working with fractions [7]. Usually the *decimals* parameter is used which means the number of decimal places of the fractional part of the token. As a rule, this parameter is chosen equal to 18. In this case, one token contains 10^{18} parts.

The approach involves the use of auxiliary units. For example, users can specify the transfer amount in thousandths or billionths of the token. Ethereum uses Finney, Gwei and other units to represent fractional parts.

In the generalized case, the entire volume of emission can be represented through one token, and its parts can be used for calculations among customers. A similar replacement can be used for comparative analysis of various blockchain-based financial instruments.

The concept will also make it possible to implement some applied algorithms. At the same time, you can use the same approaches when conducting operations with cryptocurrencies, tokens and NFTs issued in a single instance. In certain cases, the approach can simplify network interaction and transaction processing. However, difficulties might arise within the implementation of the concept due to the need for additional emission. Operations with land plots require more complex algorithms for the emission of tokens and other approaches in determining the volume of the emission.

Current algorithms do not consider the specifics of properties. A special approach for land plots is used. It considers that each of the objects is presented as a non-fungible token [8]. There is no possibility of simultaneous ownership of the plot on several levels (by the owner and sub-owners). However, such decisions are necessary to ensure ownership of the plot by a private person while this territory will belong to the state. New algorithms will make it possible to implement more complex forms of real estate ownership. There is no possibility of spliting land plots into fragments of arbitrary area and configuration or combining them in current solutions.

Non-fungible tokens can be linked to cadastral numbers [9], which include information about them. It is placed in an external database. The cadastral number determines the boundaries of the land plot and allows you to get all the information about it. All data on the territory are placed in the register. However, the database is constantly changing. Additional entries are made to it during registration of new plots. The spliting or unification of territories will be accompanied by a change in cadastral numbers. Thus, storing data on properties in a separate database creates certain difficulties when synchronizing information in the land cadaster and blockchain.

One of the features of NFTs is their indivisibility. The owner can only sell the token as a whole, as it cannot be split into parts. The widespread distribution of NFTs has led to an increase in demand for them, as well as an in-

crease in their value. The price of some tokens reaches millions of US dollars, which significantly reduces the liquidity of these assets, making them similar to precious stones, which also have a large value and cannot be divided into parts. However, the implementation of non-fungible tokens on smart contract platforms provides an opportunity to create more flexible mechanisms for issuiance and interaction with tokens.

The high prices on NFTs have led to the emergence of new tools that allow you to split tokens into parts and conduct transactions with them. The simplest solution that enables NFT splitting involves locking the token in a smart contract. At the same time, the contract issues ERC-20 tokens. When an NFT is locked, a certain number of fragments are created, for example 100. These assets can be sold for a corresponding part from the NFT's market value. The number of fragments is small, and it reduces the risk of losing one of them.

The technical implementation of fractional NFTs requires the use of additional algorithms. There are several projects that allow you to increase liquidity and split non-fungible tokens into parts, and each of the solutions has certain features.

The NFTX trading platform allows you to create repositories that hold multiple NFTs [10]. An asset lock allows contract to release a certain number of ERC-20 vTokens that represent parts of the repository. The algorithm allows the user to withdraw a random NFT from the repository by burning a single vToken.

NIFTEX and Fractional.Art platforms use more complex algorithms to split non-fungible tokens. After locking the NFT, the smart contract issues ERC-20 tokens. To trade these parts, the developers suggest creating cryptocurrency pairs on decentralized trading platforms, such as UNISWAP.

To regain ownership on a locked NFT, you must collect all the fragments and use a smart contract to return the NFT. This splitting method can lead to the fact that some fragments can be lost, burned or frozen (reserved) by their owners. If you do not know special algorithms, NFT may remain locked in a smart contract forever.

The Fractional.Art platform involves setting a certain price value for a non-fungible token. This value is determined by voting, in which only fragments' owners participate. Anyone can buy a locked NFT at an auction if they bid higher. However, the minimum value of a non-fungible token cannot be lower than the value determined during the voting. When redeeming, the fragments are burned, and their owners receive a corresponding share of the value of the non-fungible token.

The NIFTEX platform also provides the possibility of redemption of fragments. The algorithm allows any fragment owner to redeem NFT. The initiator of the transaction sets the redemption price, which requires him to block the relevant funds. If any of the other owners believe that the buyout price is low, they have the option to buy out the ownership share of the iniciator of the transaction at the specified value. At the same time, the NFT remains locked. Otherwise, the initiator receives a non-fungible token, and other co-owners get blocked funds.

Current ways of splitting NFT into fractions provide necessary functionality, but have certain limitations. Many algorithms provide only splitting into fixed parts. The technical implementation imposes certain obligations and restrictions on each of the fragment owners. In order to unlock an non-fungible token, additional social and market mechanisms are needed.

Current solutions do not consider the type of non-fungible token when splitting it into parts. If the NFT is a part of the virtual world or a flat image, the token fractions will not be associated with a specific area of that object. Prospective solutions should provide the ability to split a non-fungible token into arbitrary-sized pieces that are associated with specific parts of a digital object represented by an NFT. There is also a requirement to ensure asset ownership at multiple levels. Progressive algorithms should provide more complex forms of ownership on a virtual object, which will expand the scope of their practical application significantly.

The purpose and tasks of research

The purpose of this article is the research of specialized tools of EVM-compatible networks that implement the splitting of tokens into parts, tokenization of

properties and establishing the features of the practical application of the proposed algorithms for solving applied problems.

The research objectives include substantiation of the need for one-dimensional and two-dimensional tokens to represent linear and flat objects and conduct operations with them.

The tasks include the research of algorithms that allow entering data on the parameters of land plots directly the blockchain and the specifics of their application. The work includes the development and analysis of algorithms that ensure the splitting of objects, in particular, virtual territories into parts of an arbitrary configuration and the unification of these parts.

Tasks include the analysis of algorithms that allow ownership of territory at different levels and control over transactions by the issuer establishing the features of the practical application of the proposed solutions.

1. The concept of one-dimensional tokens

Let's consider the geometric interpretation of a one-dimensional token, the emission volume of which is one unit. It can be represented as a segment of unit length in one-dimensional space.

One of the ends of the segment is selected as a reference point. According to this interpretation, the shares belonging to the owners will be represented by parts of the segment, which are determined by the coordinates of the beginning and end. For example, Alice owns 0.45; Bob - 0.15; Carol - 0.25; Dave - 0.05; and Ed – 0.1 token (Fig. 1). Unlike ERC-20 tokens, customers own plots defined by two coordinates (for example, Bob will own plot [0.45; 0.6]). At the same time, plots equal in size will not be identical and fungible.

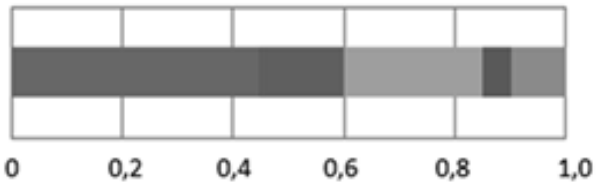

Fig. 1. A geometric interpretation of the one-dimensional token with a segment

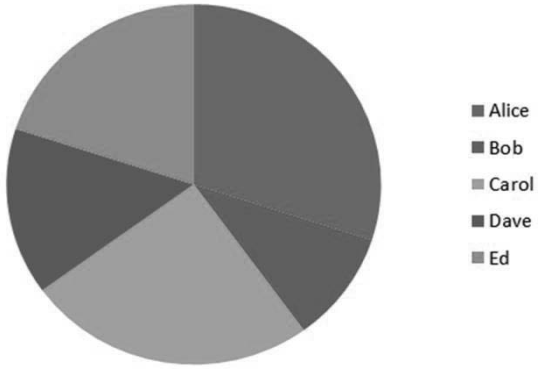

Fig. 2. Geometric interpretation of the one-dimensional token with a circle

A circle of unit length might be used for the geometric interpretation of the token instead of a line segment. It is divided into sectors of different sizes with corresponding angles (Fig. 2). It is necessary to choose a reference point on the circle, in relation to which the boundaries of the plots belonging to customers are determined. According to this interpretation, the units of measurement can correspond to radians or degrees.

The initial emission involves the transfer of token fragments to one or more customers. Fragmentation of the segment and token will be observed after performing a number of operations. After the redistribution of ownership shares, some customers may own several parts (Fig. 3). For example, Alice owns the parts [0; 0.3] and [0.65; 0.8], Bob – [0.3; 0.4] and [0.8; 1.0], and Carol – [0.4; 0.65].

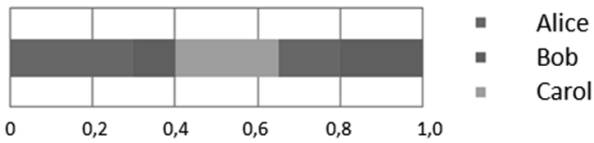

Fig. 3. Fragmentation of the one-dimensional token

The implementation of a one-dimensional token does not depend on a geometric interpretation (segment, circle or another curve). It is necessary to set its initial coordinate and its increment for each plot, which determine the share of ownership of the customer. In the case the customer has several plots, it is necessary to specify the parameters of each of them.

If only the share of ownership are to be determined, a one-dimentional token will function similarly to ERC-20 token. Each customer will own a certain part of the total emission volume. In this case, the order of location of plots, as well as their number, is not important. To find the property of the subject, it is more appropriate to use the initial coordinate of the plot and the increment of the coordinate. An alternative option is to specify the coordinates of the beginning and end of the segment directly.

Since a customer can own a large number of plots, several sources of assets can be used to perform transactions. A similar algorithm has been successfully implemented in Bitcoin [2] and other cryptocurrencies. The technical implementation of Bitcoin requires the use of one or more unspent transaction outputs (UTXO) to perform the transfer. At the same time, data on the sources of assets is included in the transaction. One-dimensional tokens can be implemented in EVM-compatible networks, and customers can send several plots in a single transaction (Fig. 4).

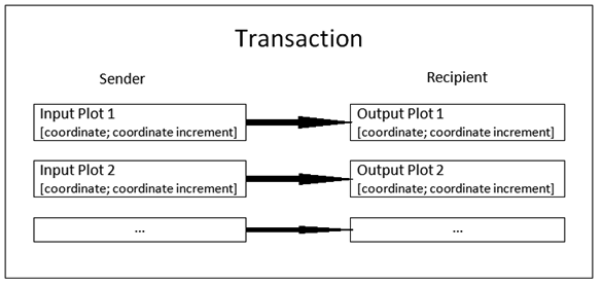

Fig. 4. Transfer of several fragments within one transaction

When transferring, the customer sends previously received plots (Input Plot 1, Input Plot 2, ...), accordingly, it is necessary to check the ownership rights to them. The transaction requires the address of the receiver and the parameters of the source and transferred plots (Output Plot 1, Output Plot 2, ...).

According to the technical implementation, sending a part of a one-dimensional token will be similar to the transfer of a ERC-20 virtual asset, but it will require two parameters that determine the beginning and end of the fragment.

2. Peculiarities of technical implementation and practical application of one-dimensional tokens

A one-dimensional token (1DT) does not conform to ERC-20 , so its implementation on an EVM-compatible network requires specialized algorithms and functions. It is recommended to choose a token emission equal to 10^{18}. If you use the value of the *decimals* parameter equal to 18, then the volume of emission will correspond to one token. In this case, the minimum plot size is limited. If we consider fragments of unit length, they can be placed sequentially from zero to the maximum value. In fact, each plot will be unique.

To implement the token in an EVM-compatible network, it is necessary to develop a smart contract and declare two arrays *Beginning_of_the_Fragment* and *Fragment_Length*, which are created with the *mapping* tool. The first array determines the coordinates of the starting point of the plot, and the second is used to specify its size (Fig. 5a).

```
mapping(address => uint[]) Beginning_of_the_Fragment;
mapping(address => uint[]) Fragment_Length;
```

a)

b)

Fig. 5. Implementation of a one-dimensional token in an EVM-compatible network

Arrays are created for each address with a non-zero balance. If the customer owns only one plot, the arrays contain one record each. In other cases, data structures include information about a large number of fragments. Elements of arrays can contain zeros that appear after complete transferring a plot to another customer.

Implementation of a one-dimensional token in an EVM-compatible network requires the use of specialized functions. The *GetSummaryBalance()* function (Fig. 6) allows you to find the total length of all fragments belonging to the customer. The algorithm adds the lengths of the fragments and finds the share

of ownership corresponding to a certain address.

Algoritm 1. Finding of the total balance.

```
1: function Get Summary Balance
2: get Number of User Fragments
3: set Summary Balance := 0,  i := 0
4:         while  i <  Number of User Fragments do
5:                   increase Summary Balance by Fragment Length[i]
6:                   increase i by 1
7:         end loop
8: return Summary Balance
```

Fig. 6. Implementation of the function *Get-SummaryBalance()*

The GetPlotsParameters() function (Fig. 7) displays two arrays of data that show the parameters of the fragments belonging to the customer. The first array represents the coordinate of the beginning of the plot, and the second denotes its length. The function does not display null array elements. The result of GetPlotsParameters() can have the following form (Fig. 5b).

Algoritm 2. Finding the parameters of the plots belonging to the user

```
1: function Get Plots Parameters
2: get Number of User Fragments
3: set i := 0
4:         while  i <  Number of User Fragments do
5:                   if Fragment Length[i] != 0 then
6:                             return Beginning of the fragment[i]
7:                             return Fragment Length[i]
8:                   increase i by 1
9:         end loop
```

Fig. 7. Implementation of the function *Get-PlotsParameters()*

To transfer a plot to another customer, you can use the function *Transfer (Sender, Recipient, Beginning of the Fragment, Fragment Length)* (Fig. 9). The procedure involves the creation of additional elements in the arrays of the recipient represented by the *Recipient* address. The resulting ownership share of the sender varies depending on which part of the fragment is being sent. At the same time, three transfer options are possible:

- The plot is transferred in full. Elements of the *Beginning of the Fragment and Fragment Length* arrays corresponding to the specified area are reset to zero.
- The sender transfer a part of the plot that is at the beginning or end of the source fragment. In this case, the algorithm change the values of array elements that determine the beginning of the fragment or its length.
- The sender transfer a fragment that divides the source plot into three parts. The first and third parts remain with the owner, and the second (middle) is transferred to the recipient. In this case, the parameters of the source fragment are changed on the sender's side. An additional element of the *Beginning of the Fragment* and *Fragment Length* arrays is also created. It includes the parameters of the third part.

The increase in the number of customers and transactions will lead to significant fragmentation of the one-dimensional token. In some cases, one customer will own several adjacent plots. If the fragments are placed in such a way that the end of one of them coincides with the beginning of the next, then it will be more appropriate to combine them. To perform this operation, the *Defragmentation*() function is used (Fig. 8). It allows you to find adjacent areas and combine them into one. Since the operation requires gas consumption and also changes the parameters of the arrays that determine the ownership of the customer, it can be carried out only by the owner.

Algoritm 4. Merging of the fragments

```
1: function Defragmentation
2: get Number of User Fragments
3: set i := 0
4:   while  i <  Number of User Fragments do
5:     set j := 0
6:         while  j <  Number of User Fragments do
7:           if (Beginning of the Fragment [i]+ Fragment Length[i] ) ==
             ==Fragment Length[j]  && Fragment Length[j] != 0 then
8:              Fragment Length[i] += Fragment Length[j]
9:           increase j  by 1
10:        end loop
11:    increase i  by 1
12:  end loop
```

Fig. 8. Implementation of the function *Defragmentation()*

Algoritm 3. Transfer of the fragment

```
1: function Transfer (Sender, Recipient, Beginning of the Fragment, Fragment Length)
2: set r := Number of Recipient Fragment
3: set s := Number of Sender Fragments
4: create Beginning of the Fragment [Recipient] [r] = Beginning of the Fragment
5: create Fragment Length [Recipient] [r] = Fragment Length
6: set i := 0
7:     while i < s do
8:         if (Beginning of the Fragment [Sender] [i] == Beginning of the Fragment then
9:             Beginning of the Fragment [Sender] [i] =Beginning of the Fragment + Fragment Length
10:        else
11:            Fragment Length[Sender] [i] = Beginning of the Fragment –
               –Beginning of the Fragment [Sender] [i]
12                 if (Beginning of the Fragment [Sender] [i] +Fragment Length[Sender] [i] !=
                   != Beginning of the Fragment+Fragment Length) then
14:                    create Beginning of the Fragment [Sender] [s]=Beginning of the Fragment +
                       + Fragment Length
15:                    create Fragment Length [Sender] [s]=Beginning of the Fragment [Sender] [i]+
                       +Fragment Length[Sender] [i] - Beginning of the Fragment - Fragment Length
16:        increase i by 1
17:    end loop
```

Fig. 9. Implementation of the function *Transfer()*

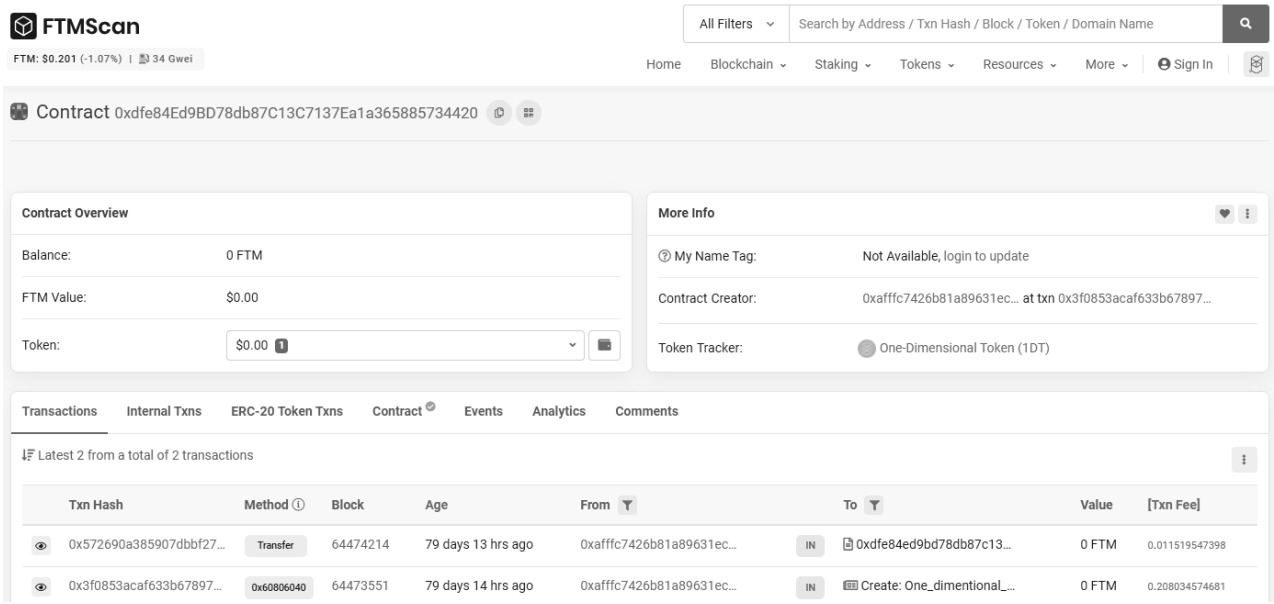

Fig. 10. One-Dimensional Token (1DT) smart contract

The One-Dimensional Token (1DT) smart contract corresponding to the proposed algorithms is deployed on the Fantom network [11] at the address: 0xdfe84Ed9BD78db87C13C7137Ea1a365885 734420 (Fig. 10).

The functions of reading the smart contract allow you to get general information about the token, determine the share of ownership and set the parameters of the plots owned by the customer. Record functions enable owners to transact and unify adjacent plots. There are also tools that provide the possibility of transferring plots using a third-party smart contract.

The plot transfer is implemented through the *transfer()* function and involves entering three parameters: the address of the recipient, the initial coordinate of the plot and its size. When transferring a plot, it is recommended to set the gas limit to more than 274000 units. Since the 1DT does not comply with the ERC-20 standard, it is possible to find out the current balance of the customer only using the corresponding reading function.

One-dimensional tokens have a number of differences compared to ERC-20 virtual assets and NFTs. These include the possibility of splitting and unifying parts of the token. At the same time, each of the fragments is unique and is defined by two parameters. The one-dimensional token allows us to demonstrate the functionality of the solution and its capabilities.

3. Algorithms for the implementation of two-dimensional tokens and their application for solving applied problems

A two-dimensional token is a virtual asset defined in a two-dimensional coordinate system. Graphical interpretation of an object – a section of a spatial surface. In some cases, it is a part of a plane, sphere or cylinder.

A token can represent an entire closed surface, allowing it to be used for a sphere. In other cases, the virtual asset corresponds to only a part of the surface. For example, a token can represent an area on a plane, bounded by a polygon. When speaking about practical application of the token, it is most appropriate to use a rectangle. This plot is splited into customer-owned fragments.

Two-dimensional tokens can be used to represent areas of a flat or spherical surface, assuming the use of identical algorithms. An arbitrary fragment belonging to the customer can be represented by a closed loop. The most optimal is the use of a polygon, the parameters of which are set by the coordinates of the vertices and the order of their sequence.

Let's consider the implementation of a two-dimensional token on a spherical surface. It is advisable to determine plot parameters in the spherical coordinate system. In this case, the position of a point in space is determined by two angles and a radius. When the radius is constant, it is enough to specify two angular coordinates.

To determine the location of the object on the surface of the Earth, we use a geographic coordinate system, and the parameters of the point are determined by latitude and longitude. For example, object coordinates can be specified in the following form: 50.449214; 30.357773. The number of characters after the separator determines the accuracy of object positioning.

A territory of arbitrary shape on a spherical surface can be represented in the form of a polygon [12]. The order of passing the vertices of the polygon determines its configuration (Fig. 11).

Fig. 11. Representation of a land plot in the form of a polygon

In certain cases, it is appropriate to represent a polygon using the coordinate of one vertex and increments that determine the position of others. The algorithm allows to reduce the amount of data recorded in the blockchain. The parameters of a plot of arbitrary shape (*Area*) can be set by an array (Fig. 12). One of the vertices is selected as the initial coordinate (*Initial coordinate*). The coordinates of other points are set in the form of increments in relation to the initial coordinate (*coordinate increment 1, coordinate increment 2, ... coordinate increment n*). The

Area

	Latitude	Longitude
Initial coordinate	**50,449214**	**30,357773**
coordinate increment 1	0,004703	0,113006
coordinate increment 2	-0,123239	0,04289
coordinate increment 3	-0,120425	-0,131187
coordinate increment 4	-0,075794	-0,213764
coordinate increment 5	-0,145759	-0,375646
coordinate increment 6	-0,090647	-0,464117
coordinate increment 7	-0,048376	-0,380012
coordinate increment 8	0,043858	-0,466657
coordinate increment 9	0,089249	-0,234732
coordinate increment 10	0,145127	-0,288953
coordinate increment 11	0,163612	-0,175346

Fig. 12. Representation of plot parameters using an array

order of the vertices when passing a closed circuit corresponds to clockwise movement.

The algorithm allows you to perform operations with fragments of an arbitrary configuration, to split and unify them. A special configuration area may contain a void if its interior belongs to another customer. In this case, the territory can be represented by two polygons that do not contain cavities, but have common edges and vertices.

Two-dimensional tokens can represent areas of any flat or spherical surfaces. They can be used in virtual worlds containing territories owned by customers. The developed algorithms can be used to represent areas of the Earth or another celestial body.

Two-dimensional tokens can be used to distribute NFTs or other virtual assets among multiple owners. At the same time, each customer can own a flat fragment of any configuration.

If it is necessary to have sub-owners of the first and second levels, then the process of emission and transfer of the land plot will be different. The first level sub-owner must issue a token. In order to carry out operations with a land plot, a second-level sub-owner must initiate a transaction. However, to implement it confirmations from sub-owners of the first and second levels are required. When using such an algorithm, all transactions with virtual assets will be controlled by all their owners.

When transferring, information about the parties will not be disclosed, since only their addresses are placed in the distributed database. Blockchain technologies will provide protection against unauthorized access, eliminate the possibility of canceling a confirmed transaction, and also prevent double spending operations [14].

When creating and conducting operations with large territories that include a significant number of vertices of the bounding polygon, a significant amount of information must be entered into the blockchain. We can reduce the amount of data by using a base coordinate and coordinate increments for the remaining vertices. It is advisable to use abbreviated coordinates within operations with small areas.

The use of two-dimensional tokens for the tokenization of land plots will require interaction with databases containing cadastral numbers and other information about properties. It is more appropriate to place the specified information in decentralized databases, e.g. by using IPFS.

Conclusions

One-dimensional tokens are a powerful tool for testing, creating arbitrary-sized plots and conducting transactions. The developed smart contract allows customers to own fragments of any one-dimensional object and perform operations with them.

2D tokens can be used to fragment maps, flat images, and other 2D objects. They have a number of advantages over NFTs when used to represent land plots and virtual territories. Two-dimensional tokens allow you to create fragments of arbitrary configuration, split and unify them.

The developed algorithms can be used to represent the territories of virtual worlds. Current projects use fixed-sized plots, while two-dimensional tokens provide greater flexibility by allowing fragments of arbitrary configuration to be operated on.

The technology can be applied to spherical surfaces, including the Earth surface, but it requires setting a reference point. The use of two-dimensional tokens will require interaction with external databases containing data on cadastral numbers. At the same time, the algorithm must be compatible with blockchain-oracles that provide access to external state databases that personalize the owners of tokenized land plots.

Two-dimensional tokens can provide ownership of a surface area at different levels. A certain fragment can have one owner who is the issuer and controls the operations with the virtual asset and a sub-owner who conducts transactions with the plot or its parts directly. Such an algorithm provides an opportunity to implement a model of ownership that is characteristic of the land market of an individual country.

References

1. Bauer, D.P. (2022). Ethereum Name Service. In: Getting Started with Ethereum. Apress, Berkeley, CA. doi: 10.1007/978-1-4842-8045-4_9.

2. Lisdorf, A. (2023). Bitcoin. In: Still Searching for Satoshi. Apress, Berkeley, CA. doi: 10.1007/978-1-4842-9639-4_4.
3. Arslanian, H. (2022). Ethereum. In: The Book of Crypto. Palgrave Macmillan, Cham. doi: 10.1007/978-3-030-97951-5_3.
4. Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, Ian Norden, Abdelhamid Bakhta, "EIP-1559: Fee market change for ETH 1.0 chain," Ethereum Improvement Proposals, no. 1559. (2019). [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-1559.
5. Jethmalani, Mehak, & Yaskil, Rivka, Aspler. (2023). Non-fungible token (NFT) purchase and transfer system. US Patent US20230298001, filed March 21, 2022, and issued September 21, 2023.
6. Fabian Vogelsteller, Vitalik Buterin, "ERC-20: Token Standard," Ethereum Improvement Proposals, no. 20. (2015). [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-20.
7. Ritesh, Modi. Solidity Programming Essentials. (2018). Birmingham, UK: Packt Publishing, 2018. ISBN 9781788831383.
8. Shang, Qiuyun & Price, Allison. (2019). A Blockchain-Based Land Titling Project in the Republic of Georgia: Rebuilding Public Trust and Lessons for Future Pilot Projects. Innovations: Technology, Governance, Globalization. 12. 72-78. doi: 10.1162/inov_a_00276.
9. Dale, Peter, & John McLaughlin. (2020). Land Administration (Oxford, 2000; online edn, Oxford Academic, 12 Nov. 2020). doi: 10.1093/oso/9780198233909.001.0001.
10. Introducing Our Decentralized NFT Marketplace. https://medium.com/nftx/introducing-our-decentralized-nft-marketplace-1c1c4d394724
11. Fantom Foundation: https://fantom.foundation/
12. Cadle, Farris W. (1991). Georgia Land Surveying History and Law. University of Georgia Press. ISBN 978-0-8203-1257-6.
13. Bellare, Mihir, & Neven, Gregory. (2006). Identity-Based Multi-signatures from RSA. Topics in Cryptology – CT-RSA 2007. pp. 145–162.
14. Chohan, Usman W. (2021). The Double Spending Problem and Cryptocurrencies. doi: 10.2139/ssrn.3090174.

Received: 01.12.2023

Про авторів:

Струтинський Сергій Васильович,
доктор технічних наук, професор кафедри прикладної гідроаеромеханіки і механотроніки Національного технічного університету України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій в українських виданнях – понад 200.
Кількість наукових публікацій в зарубіжних виданнях – понад 10.
Індекс Хірша – 5. https://orcid.org/0000-0001-9739-0399

Яланецький Валерій Анатолійович,
старший викладач кафедри інформаційних систем та технологій Національного технічного університету України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій в українських виданнях – понад 30.
https://orcid.org/0000-0001-6163-0258

Місце роботи авторів:

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»,
проспект Берестейський 37,
Тел.: (044) 204-82-54,
E-mail: strutynskyi@gmail.com,
v.yalanetskyi@gmail.com

УДК 681.3 http://doi.org/10.15407/pp2024.01.048

Д.В. Рагозін, А.Ю. Дорошенко

КЛАСИФІКАЦІЯ НИЗЬКОЧАСТОТНИХ СИГНАЛІВ ЗА ДОПОМОГОЮ МЕТОДІВ КЛАСТЕРИЗАЦІЇ

У статті розглянуто методи класифікації сигналів звукового та інфразвукового діапазону за допомогою алгоритмів кластеризації у випадках, коли присутня лише загальна апріорна інформація, наприклад, невідомі типи об'єктів, які генерують відповідні сигнали. Розглянуто підготовку даних звукового діапазону, особливості первинної обробки набору даних, параметри вибору алгоритму залежно від особливостей набору даних. Подано приклади кластеризації набору даних за допомогою алгоритму OPTICS, описано можливості каскадної обробки набору даних.

Ключові слова: класифікація даних, навчання без контролю, кластеризація даних, обробка звуку.

Вступ

Сучасні методи обробки сигналів часто зосереджені на застосуванні нейронних мереж. Однак нейромережеві методи мають значний недолік – велику енергоємність, що у випадку достатньо довгого аналізу є фактором, який обмежує застосування нейромереж. Також застосування певних типів нейромереж обмежено у разі необхідності попереднього навчання нейромережі. Тому для практичних застосувань цікаво проаналізувати можливості застосування методів, альтернативних нейромережам і оцінити їхню ефективність у вирішенні задач класифікації сигналів низьких частот. У статті ми розглянемо особливості задач класифікації сигналів, особливості обробки звукових сигналів, сучасні методи аналізу сигналів, практичне застосування методів і можливості подальшого їхнього розвитку.

Обробка низькочастотних сигналів

Зазначимо, що низькочастотні сигнали не обмежуються електромагнітними хвилями, і обробка низькочастотних електромагнітних хвиль є обмежено корисною, оскільки для практичної передачі інформації в абсолютній більшості випадків використовуються високі частоти. Звукові та інфразвукові хвилі несуть у собі значну кількість інформації, яка може бути проаналізована в автоматичному режимі і в ощадливому режимі споживання енергії. Останнє важливо у разі автономної роботи датчиків, які додатково можуть і не випромінювати електромагнітну енергію.

Важливо, що на відміну від «домену» електромагнітних хвиль, де маємо апріорну інформацію щодо спектру сигналу і певних сигнатур сигналу, апріорна інформація щодо звукового сигналу зазвичай відсутня. Під сигнатурою тут ми розуміємо типові співвідношення потужностей сигналу в піддіапазонах виділених частот і їхню типову зміну з плином часу. Це дуже грубе визначення, оскільки багато факторів впливає на конкретні вимоги до аналізу спектру. Детальніше з питаннями симуляції сигнатур можна ознайомитися в [1], де розглядається один із варіантів протоколу Wi-Fi, який використовується як база для великого сімейства протоколів передачі даних. Отож, під час аналізу сигналів електромагнітного спектру ми знаємо з чим стикаємось, можемо проаналізувати нижні рівні комунікацій у визначеннях моделі OSI [2] і певним чином скористатися отриманою інформацією. Наприклад, у практичній площині розпізнавання об'єктів або радіоелектронної боротьби. Описаний аналіз електромагнітного спектру стає можливим, оскільки використання діапазонів електромагнітного спектру формалізоване. Однак у випадку, коли об'єкт не випромінює електромагнітні хвилі (режим радіомовчання), необхідно застосовувати інші методи. У звуковому діапазоні така стандартизація неможлива, окрім випадків штучного обмеження потужності генерації звука під час роботи якогось обладнання. Тому після пе-

рвинного аналізу звукового спектру ми маємо скористатися одним із методів навчання без контролю (англійською мовою unsupervised learning), найбільш актуальні з яких розглянуто в [3]. Зокрема, специфічні для промислових завдань методи навчання без контролю розглянуті в [4].

Нашою метою є класифікація об'єктів, які створюють певну електромагнітну або звукову картину до певних типів за умови, що апріорно ми не знаємо, які у нас типи об'єктів, чи з'являються у нас нові об'єкти. Далі будемо також позначати таку класифікацію терміном «кластеризація».

Розглянемо спочатку первинну обробку звукового сигналу, потім вторинну. Тобто спочатку методи, які дозволяють редукувати розмірність даних з метою подальшої ефективної обробки і кластеризації редукованих даних.

Первинна обробка сигналу

Звісно, первинна обробка низьких частот проводиться шляхом частотного аналізу, але є декілька можливостей такого аналізу. Практичні частоти для аналізу звукової інформації починаються від 0.1 Гц і закінчуються на 30 кГц (ультразвук). Таким чином діапазон обробки складає до 18 октав (під октавою ми розуміємо діапазон частот від f до $2f$ Гц). Стандартизовано у промисловій обробці потужність хвиль вимірюється у частинах діапазону у 1/3 октави (піддіапазонах). Для стандартного звукового діапазону 20 Гц - 20 кГц (10 октав), потужність сигналу вимірюється у 30 піддіапазонах у третю частину октави. Зазначимо, що ця технологія вимірювання є практичним спадком часів, коли частотний аналіз виконувався за допомогою фільтрів смуг сигналу шириною у третю частину октави на стандартизованих частотах. Зараз обчислювальна потужність збільшилась і дозволяє проводити Фур'є аналіз спектру, як швидкий, так і звичайний, або їхню комбінацію. Кінцевою метою аналізу є отримання векторів $p_i = (p_0, p_1, \ldots, p_n)$, де p_i є потужністю хвиль для певного піддіапазону. Послідовність таких векторів далі може використовуватися для класифікації.

Порівняно з електромагнітними хвилями, звуковий діапазон є важчим для аналізу, оскільки велика кількість діапазонів може мати сигнатури незалежних сигналів.

Потрібно зробити ремарку, що з практичних міркувань необов'язково аналізувати весь діапазон від 0.1 Гц до 30 кГц. Звукові хвилі починаються з 5 Гц, вібрацію практичніше вимірювати до 50 Гц, у підсумку це залежить від типу вимірювань і встановлення датчиків. Далі дискретне перетворення Фур'є для певного сигналу дає нам лише певне уявлення про сигнал, його характерну потужність у діапазоні, оскільки дискретне перетворення Фур'є вимірюється для значень частоти, кратних частоті дискретизації. Тобто, якщо перетворення Фур'є дає нам амплітуду і фазу для частот $(f_0, f_1, f_2, \ldots, f_{n-1}, f_n)$, то різниця сусідніх частот $(f_i - f_{i-1})$ завжди однакова. Проте у природному середовищі практично ніколи не зустрічаються частоти, які точно співпадають з будь-яким f_i, тому на вихідному графіку амплітуд частот ми бачимо не пікові значення частоти, а дзвоноподібні хвилі (див. рис. 1). Для точного вимірювання частоти сигналу можна використати дробове перетворення Фур'є, яке може визначити амплітуду і фазу для будь-якої частоти. Але з міркувань обчислюваної потужності це неефективно і практично неможливо без попереднього аналізу сигналу. Тому піддіапазони в третину октави 1) добре усереднюють сигнал; 2) розносять гармоніки сигналу у різні піддіапазони. Далі на базі піддіапазонів можливо проводити вторинну обробку сигналу.

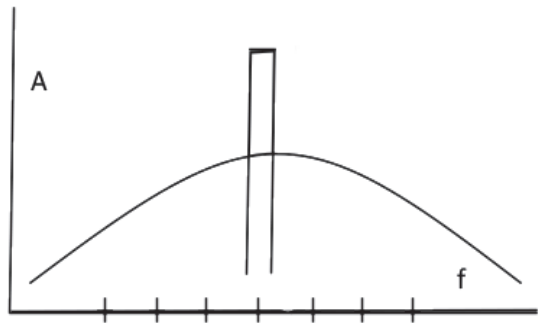

Рис. 1. Дзвоноподібний спектр сигналу у порівнянні з сигналом, частота якого співпадає з однією з частот сітки перетворення Фур'є.

Вторинна обробка сигналу

Для вторинної обробки сигналу з n піддіапазонами використовуємо вектор середніх значень амплітуд для кожного піддіапазону $a^n = (a_1, a_2, \ldots, a_n)$. Фазова інформація відкидається, бо для неї неможливо знайти «середнє» значення. Оскільки сигнал постійно змінюється, весь час спостережень за сигналом має бути розбитий на певні характерні інтервали залежно від часу протікання процесу, за яким ми спостерігаємо. Наприклад, у разі прихованого спостереження за рухом людей або техніки на дорозі можна спостерігати за сигналом і робити усереднення протягом інтервалів 5-7 секунд. У випадку великих приміщень інтервал може бути 15 секунд. Таким чином сигнал від спостережень є послідовність векторів $\{ a^n \}$, яку вже можна класифікувати.

Як було зазначено вище, у нас немає апріорної інформації про спектр сигналів, які спостерігаються, тому ми мусимо зробити певні припущення щодо вигляду класифікованого сигналу. На рис. 2 представлені приклади наборів даних $\{ a^2 \}$ (двовимірні) у поясненнях до пакету класифікації Scikit [5], заради ясності зроблені лише двовимірними. У більшості випадків аналізу розмірність сигналу перевищує 10.

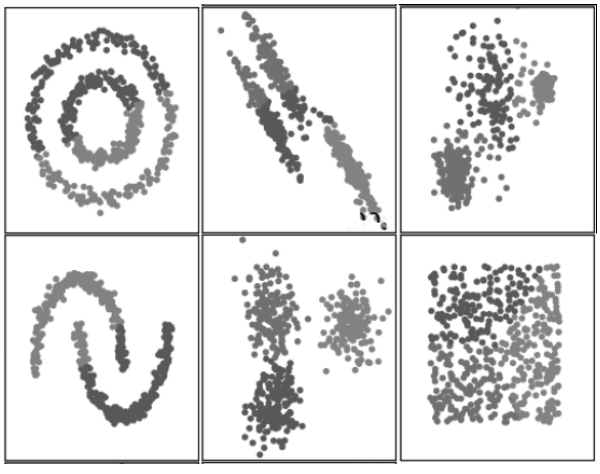

Рис 2. Різні типи наборів даних для кластеризації (приклад з бібліотеки SciKit).

Згідно експериментальних даних, різні об’єкти (легкові автівки, вантажні, важка бронетехніка, пішоходи, тварини середнього розміру, промислове обладнання, побутові прибори) дають (інфра)звукову картину певного сорту, яка локалізована в певних частотах, тобто не дає шумоподібних сигналів або сигналів у широкій смузі частот. Так автор спостерігав сигнал від геофона, спричинений рухом танка Т-64 на швидкості 20 км/год на відстані у 50 м, і це була синусоїда однієї частоти і практично однакової амплітуди, яка на графіках з рис. 2 виглядатиме як крапки, сконцентровані практично в одному місці. У підсумку зазначимо, що отримані набори даних виглядатимуть як правий у верхньому ряду та як середній у нижньому ряду на рис 2, що дозволяє оцінити, які властивості алгоритму класифікації мають значення для подальшого аналізу, а які – ні. Зазначимо, що у первинному сигналі досить багато шуму і набір даних переважно має сигнатури шумів, і тут ми розглядаємо набір даних, вже очищений від шуму, методи «очистки» від шуму розглядаються нижче.

Отож, ми вважаємо, що кількість типів об’єктів, які генерують звукові хвилі, обмежена (автівки, вантажівки, тяжка бронетехніка, пішоходи, велосипедисти, мотоцикли, тварини), також можуть бути інші події (далекі або близькі різкі звуки), які на рис. 2 мали б вигляд хаотично розташованих крапок. Останні мають класифікуватися окремо. Звісно, нейромережа могла б акуратніше класифікувати події, але не завжди є можливість залучити канал зв’язку або обчислювальні потужності чи коректно розробити нейромережу для всіх можливих випадків.

На рис. 3 показана типова картина звукових шумів з одного з випадків розглянутих у [6]. Якщо придивлятися до лівої частини рисунку, ми побачимо на темному тлі більш вертикальні світлі смуги, які позначають відносно більшу спектральну щільність енергії сигналу на певних частотах, а відносно перебігу часу (вертикальна вісь) ми бачимо, що частотна картина зберігається у часі. Далі робиться підсумок енергії по піддіапазонах, отримується вектор амплітуд a^n, як описано вище, і результат передається до класифікатора. Виходячи з наявних і описаних вище обмежень, ми класифікуватимемо звуки за допомогою методів кластеризації, а також опишемо особливості цих методів і далі наведемо приклади кластеризацій.

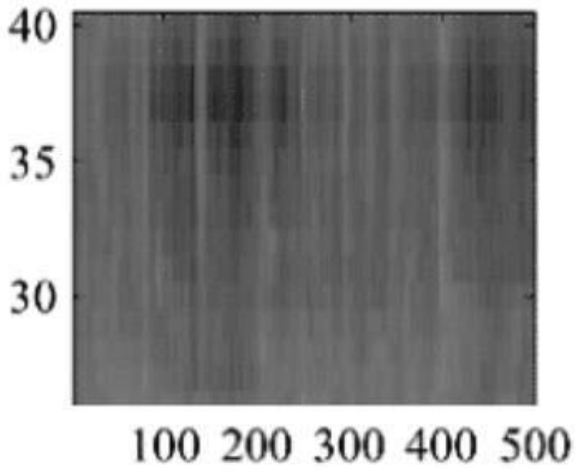

Рис 3. Приклад картини шумів у випадку декількох груп людей, які одночасно розмовляють (горизонтальна вісь – частота, вертикальна – час).

Методи класифікації за допомогою кластеризації

Для подальших експериментів було вирішено використовувати наявну алгоритмічну базу бібліотеки Scikit[5], яка на даний момент має найбільший алгоритмічний доробок методів класифікації.

Одним із найперших і найпростіших є метод K-means [7], який розподіляє точки набору даних на визначену кількість кластерів. Але у більшості випадків метод є дуже простим, має мінімальну параметризацію, тому досить швидко були розроблені покращені методи кластеризації. Розглянемо спочатку можливості параметризації таких методів.

Кількість кластерів. У нашому випадку досить часто виявляється, що ми не знаємо кількість кластерів, тобто кількість типів об'єктів, що будуть розрізнені в наборі даних. Кількість кластерів задається або певним числом, що має наслідок подальшої кластеризації об'єктів на більшу кількість типів, або автоматично. Автоматична кластеризація вимагає більшу обчислювальну потужність, оскільки дуже часто алгоритм вибудовує навколо набору даних потужні допоміжні структури даних. Кількість кластерів може задаватися неявно. Наприклад, мінімальна дистанція між точками одного кластеру (найпростіше), мінімальна кількість точок у кластері, або мінімально відносна щільність даних у кластері відносно загальної щільності даних. Окремо може задаватися метрика аномалій, «зайвих» точок, які не потрапляють у кластери, а вважаються або за визначенням нечастими аномаліями, або похибками вимірів.

Масштабованість. Основними її параметрами є кількість точок у наборі даних та кількість кластерів. Це впливає на швидкодію алгоритму та обсяг пам'яті й може бути фактором, який лімітує використання методу.

Типове використання. Є кластеризація загального призначення, є кластеризації для невеликої (десяток) або великої (сотні) кількості кластерів. Вирізняються методи, які краще працюють у випадку багатовимірних даних, оскільки використовують структури даних для акселерації доступу до масиву точок, як-от KD-дерев [8]. Можливі обмеження на зв'язки між кластерами, різні умови для визначення аномалій, введення ієрархії кластерів (що можливо за використання тих же KD-дерев [8]), задання різної щільності кластерів, індуктивне формування кластерів.

Метрика визначення кластерів. Звичайна евклідова відстань між точками, використання графів сусідів, лімітована сусідами метрика відстані між точками, відстань типу Махаланобіс.

Перед тим, як перейти до найбільш цікавих методів, підсумуємо й зробимо зауваження щодо параметризації. Звісно, методів менше, ніж усіх комбінацій параметрів кластеризації, оскільки виникнення нових методів стимулювалося необхідністю оптимізувати наявні методи для великих наборів даних. Великі обсяги даних вимагають швидкої індексації наявних даних, тому певні методи виникли як результат інтеграції простих методів кластеризації та програмних структур, що є акселераторами доступу до даних, таких як, KD-дерево [8]. Останнє вирішує важливу задачу пошуку сусідів точки у багатовимірному просторі з логарифмічною складністю, а без наявності KD-дерева пошук сусідів стає дуже дорогим за ресурсами завданням. У підсумку – для вибору ефективного алгоритму кластеризації необхідно мати апріорну інформацію щодо вхідних даних. Для звукового ді-

апазону вистачає тижня запису даних і подальшого візуального аналізу.

Окремо зазначимо, що певна кількість алгоритмів має імплементації лише в рамках бібліотек мовою Python, що може викликати погіршення швидкодії від 3 до 10 разів. Є імплементації багатьох алгоритмів від приватних осіб, але, зважаючи на складність внутрішніх структур даних, не можна стверджувати, що такі алгоритми відтестовані. Це накладає обмеження на обсяги даних, які можуть бути оброблені в реальному часі. Зупинимося на декількох алгоритмах.

Найбільш простим алгоритмом є K-means [7]. Оскільки історично він був розроблений раніше за інших, він є найпростішим і має найбільше недоліків. Основним практичним недоліком є те, шо неможливо відразу за апріорними даними встановити необхідну кількість кластерів. У випадку, коли кількості кластерів замало, декілька типів об'єктів можуть бути віднесені до одного кластеру – тобто ми їх не зможемо відрізнити. А якщо кластерів забагато, об'єкти одного типу будуть віднесені до різних кластерів, що спотворює аналіз. Це показано на рис. 7. Можна проаналізувати відстань між отриманими кластерами і, якщо відстань між геометричними центрами деяких кластерів менша за середнє значення, – зменшити кількість кластерів і перезапустити кластеризацію. Але такий метод погано працює у зворотньому напрямку – коли кластерів замало. Ускладнена й евристика, яка дозволяє зрозуміти, чи то дійсно мала геометрична відстань між класами, чи то така особливість набору даних. Ці обмеження привели до розробки більш досконалих методів. Наприклад, mean-shift [9], де параметризувалася необхідна щільність кластерів та мінімальний об'єм простору для формування кластеру. Кластеризація уточнюється ітеративно, водночас змінюється геометричний центр кластеру, тому цей метод ускладнений для паралелизації на відміну від K-means. Взагалі «покращені» методи відрізняються підвищеним споживанням пам'яті та обчислювальних ресурсів.

Покращити кластеризацію намагалися за допомогою введення ієрархії кластерів [10], що може бути корисним, якщо ми знаємо, що набір даних має ієрархічні властивості – тобто різні типи об'єктів можуть генерувати дуже схожі спектри, але це працює лише у незначній кількості випадків.

Можна сказати, що метод DBSCAN [11] має революційні відмінності від [7],[9],[10] оскільки орієнтується на щільність точок набору даних у геометричному просторі і кластеризує дані за принципом знаходження областей простору з найбільшою щільністю даних. Тобто вимагає щоб для елемента даних на певній дистанції ε від нього було не менше s_{min} точок даних (*samples*). Таке визначення дозволяє мати кластери різної форми, не обов'язково опуклі. Якщо певні точки набору даних згідно з обмеженнями s_{min} і ε не можуть бути співставлені до певного кластеру, то вони вважаються outliers або називатимемо їх аномаліями. Зазвичай аномалії розташовані в областях простору, де немає великої щільності даних і з фізичної точки зору, – оскільки ми розглядаємо сигнали – звичайно є перешкодами. На рис. 4 – що взятий із коментарів до SciKit – аномалії зображено невеликими чорними колами, їхній алгоритм не зміг долучитися не до жодного з трьох кластерів.

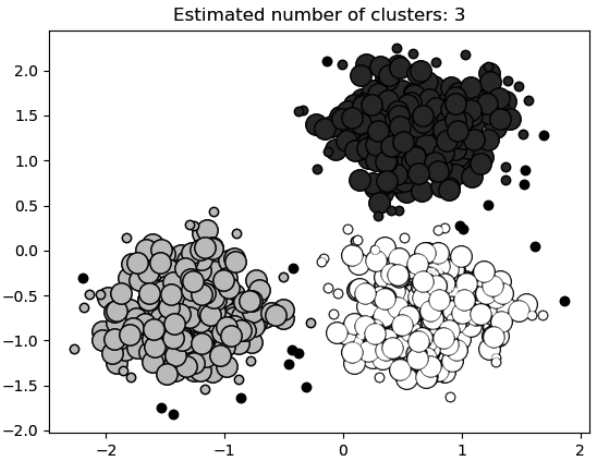

Рис. 4. Приклад кластеризації з аномаліями

Кількість аномалій регулюється вищевказаними параметрами s_{min} і ε, і кількість аномалій може бути регульована від 1% до 5% від загального обсягу даних. Така параметризація важлива у випадках, коли ми намагаємося класифікувати аномалії, і

подальша класифікація залежить від правильного визначення аномалій.

Із недоліків алгоритму відмітимо досить великий обсяг необхідної пам'яті. Розвиває ідеї DBSCAN алгоритм OPTICS [12], який дозволяє мати кластери з різними ε, і додає поняття досяжності, що фактично у порівнянні з DBSCAN дозволяє брати для кластерів не одне значення ε, а цілий інтервал значень $[\varepsilon_{min},\varepsilon_{max}]$. На додачу OPTICS оптимізований щодо обсягу використаної пам'яті.

Останнім із цікавих алгоритмів розглянемо BIRCH [13], який може оперувати даними, що не вміщуються в пам'яті. Кластеризація здійснюється на основі суб-кластерів, до яких намагаються віднести всі значення набору даних, а далі вже комбінувати су-кластери для досягнення необхідної кількості кластерів.

Методи [7],[8],[9],[10],[11],[12],[13] були розроблені для певних задач, для певних обмежень (серед яких і обсяг пам'яті, і швидкодія) і для певних апріорних знань про вхідні дані. Зазначимо, що серед усіх методів кластеризації неможливо виріznити найкращі і найгірші, оскільки всі вони орієнтовані на певні дані.

Якщо аналізувати наші апріорні дані – звуковий або електромагнітний спектр, то найбільш цікавим кандидатом виглядає алгоритм OPTICS [12], який ми далі використаємо для кластеризації.

Збір даних

Найпростішим варіантом збору даних є звичайний мікрофон і звичайний (офісний) комп'ютер. Цього обладнання вистачає для запису звукових сигналів у діапазоні 20Гц – 10кГц, також необхідний звуковий редактор для запису звуку. Дещо більший діапазон частот буде за умови використання напівпрофесійних або скерованих мікрофонів і підсилювача. У випадку вібраційних хвиль можна використати геофони (geophone), які продаються у маркетплейсах електроніки. Необхідним буде також підсилювач електричного сигналу, який може видати сигнал на лінійний вхід звукової карти.

Для підготовки набору даних для подальшої обробки із зроблених записів звукового діапазону можна використати Matlab або Python, оскільки ці середовища програмування мають найбагатші бібліотеки обробки сигналів. Сигнал ділиться на однакові інтервали (наприклад 10 секунд), довжина інтервалу може змінюватися відносно типу об'єкту спостереження. Для ділянок доріг може бути 10 с, для приміщень або складів – 15-20 с. Отримані інтервали сигналу обробляються за допомогою швидкого перетворення Фур'є (FFT), водночас нам достатньо амплітудно-частотної характеристики (АЧХ). Далі отримана АЧХ розділяється на піддіапазони, у кожному піддіапазоні амплітуди усереднюються до одного значення амплітуди. Для кожного інтервалу отримаємо вектор амплітуд $a^n = (a_1, a_2, \ldots, a_n)$, який буде елементом набору даних. Для інтервалу у 10 секунд щодоби матимемо 8640 векторів, і це число впливає на вибір метода кластеризації. Для аналізу добових даних бажано мати набори даних, що перекриваються у часі. У найпростішому випадку можна дробити добовий набір даних на множини A^D_A, A^D_P, де D – умовний номер доби, а A і P позначає першу і другу половини доби. Далі для кластеризації можна формувати набори (A^D_A, A^D_P), (A^D_P, A^{D+1}_A), (A^{D+1}_A, A^{D+1}_P), (A^{D+1}_P, A^{D+2}_A),… У разі наявності обчислювальних потужностей та пам'яті можна формувати набори (A^D, A^{D+1}), (A^{D+1}, A^{D+2}), (A^{D+2}, A^{D+3}), … або навіть довші набори. Далі перейдемо до дослідів із кластеризації.

Результати кластеризації

Для дослідів із кластеризації був обраний метод OPTICS [12], згідно з розмірковуваннями і обчисленнями обсягів набору даних, які подані у статті вище. Використовувалася бібліотека Scikit [5] для імплементацій методу OPTICS, довжина векторів набору була від 10 до 20. Нижче для ілюстративних випадків використана довжина вектору в 10, оскільки більша довжина вектору лише захаращує ілюстративні приклади. Горизонтальна вісь показує середню частоту піддіапазону, вертикальна вісь показує відносну потужність сигналу. Абсолютне зна-

чення потужності сигналу не має жодного значення для методів кластеризації. На рис. 5 показаний результат кластеризації для одного з вібраційних наборів даних.

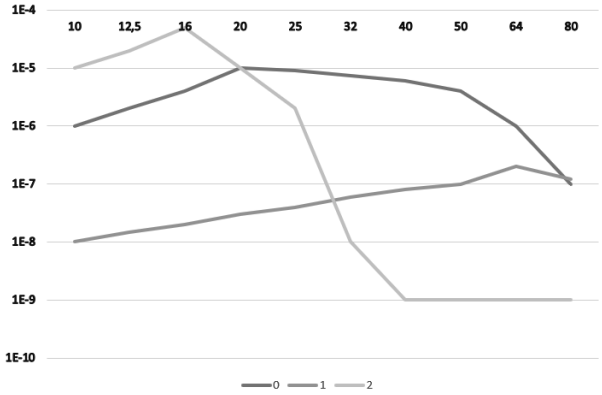

Рис. 5. Приклад коректної кластеризації на три кластери (OPTICS).

На рис. 5 ми бачимо що алгоритм визначив 3 кластери. У першому кластері (0) пік потужності зареєстрований на частоті 20 Гц і досить плавно падає до 50 Гц. Пік другого кластеру (1) припадає на 16 Гц і швидко спадає після 25 Гц. Третій кластер (2) представлений високочастотним шумом. Окремо зазначимо, і це важливо, що на частотну картину накладаються особливості мікрофона або поверхні, на якій стоїть геофон, у різних випадках вимірювань конкретні значення потужності можуть відрізнятися, але якісна картина співвідношення потужностей сигналу у кластерах залишається незмінною. На наступному рисунку 6 показаний ще один приклад роботи алгоритму. Перший кластер (0) має пік на низьких частотах і поступово спадає в області високих частот, другий кластер (1) має пік в області 40 Гц, третій кластер (2) має пік в області 16 Гц.

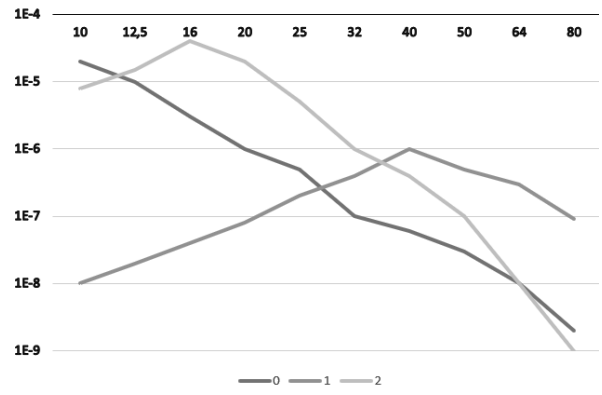

Рис. 6. Інший приклад коректної кластеризації для трьох кластерів.

На рис. 7 показаний один з перших дослідів роботи з простішими алгоритмами на базі K-means. На рисунку показаний приклад некоректної кластеризації.

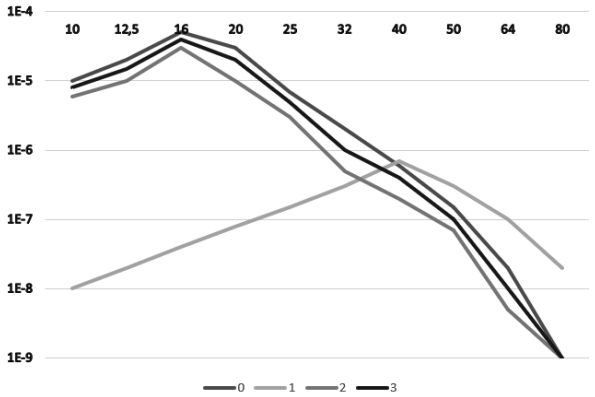

Рис. 7. Приклад некоректної кластеризації під час використання методу K-means.

Методу кластеризації, який орієнтований на наперед задане число кластерів, було задано перебільшену кількість кластерів – 4. Реальна кількість кластерів (визначена *a posteriori*) – 2. Таким чином алгоритм штучно «розтягнув» один із кластерів, у якого відносно більша кількість елементів, на три кластери (0,2,3). У разі, коли на цьому ж наборі даних алгоритмові пропонується розподілити дані на два кластери, лишаються кластери, наприклад, тільки 1 і 3.

Із практичного боку маємо ще декілька спостережень щодо підготовки даних до кластеризації. По-перше, не обов'язково мати діапазони в 1/3 октави, кластеризація добре працює і у випадку ½ октави. Більшість характерних шумів від об'єктів є синусоїдальними сигналами небагатьох частот, від 3 до 5, тому діапазон в 1/3 октави можна вважати оптимальним. Важка техніка може давати синусоїду лине однієї частоти, можливо тому, що сильні вібрації підвіски екранують інші частоти.

Маніпуляції з виділення піддіапазонів «низької», «середньої» та «високої» частоти шляхом простого ділення піддіапазонів на три групи, потім - з кластерізаціями у таких підгрупах, не дають нам переваг перед прямим використанням кластеризації. Такий висновок можна зробити як мінімум з тих засад, що доволі важко пояснити фізичний смисл такого вибору.

Алгоритми DBSCAN [11] та OPTICS [12], і не лише вони, дають нам також елементи набору даних, позначені як «без кластеру», або аномалії. Зазвичай це точки набору даних, які знаходяться далеко від геометричного центру кластеру, максимальна відстань залежить від описаних вище параметрів алгоритмів. Зважаючи на фізичний сенс набору даних, аномалії з більшими амплітудами на високих частотах можуть з великою вірогідністю вважатися шумом, а аномалії на низьких частотах можуть бути додатково розглянуті щодо їхніх джерел.

Каскадовані кластеризації

Для аналізу наборів даних необхідна їхня правильна підготовка, оскільки, залежно від фізичних особливостей середовища (вібрації дороги і вібрації цеха дуже різні), необхідно певним чином виділяти цікаві нам набори даних,. Сучасні алгоритми кластеризації, які можна знайти у [5], пропонують як кластеризацію, так і виділення аномалій. І дуже часто цікаві для аналізу сигнали на фоні первинного набору даних виглядають аномаліями. Тому важливе визначення каскадування кластеризацій, наприклад, на першому етапі виділення змістовних аномалій, а на другому етапі їхня кластеризація. На третьому етапі також можлива кластеризація аномалій, що залишилися після першої кластеризації. Конкретне каскадування дуже сильно залежить від об’єкта, аналіз якого ми проводимо, і конкретні поглиблені схеми каскадування повинні розглядатися виключно для своєї предметної області.

Висновки

У статті розглянуто можливості, найбільш цікаві алгоритми кластеризації і приклади роботи з набором даних для аналізу частотної характеристики сигналів звукового діапазону. Зокрема, розглянуто «живі» приклади аналізу сигналів вибраними методами кластеризації для звукового і вібраційного діапазонів з поясненням результатів. У сучасному світі результати таких аналізів використовуються для спостереження за користуванням об’єктами і для аналізу об’єктів для провадження певних видів промислової діяльності, для яких необхідно регулювати режими шуму і вібрацій.

References

1. F. Meneghello, N. Dal Fabbro, D. Garlisi, I. Tinnirello, M. Rossi. A CSI Dataset for Wireless Human Sensing on 80 MHz Wi-Fi Channels. IEEE Communications Magazine, 2023. https://doi.org/10.48550/arXiv.2305.03170
2. ISO/IEC 7498-1:1994 Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model. June 1999.
3. Liu, Y., Li, J. (2023). A Survey of Spectrum Sensing Algorithms Based on Machine Learning. // In Proc. Xiong, N., Li, M., Li, K., Xiao, Z., Liao, L., Wang, L. (eds) Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery. ICNC-FSKD 2022. Lecture Notes on Data Engineering and Communications Technologies, vol 153. Springer, Cham. https://doi.org/10.1007/978-3-031-20738-9_97
4. M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. H. Rohban, and M. Sabokrou, “A unified survey on anomaly, novelty, open-set, and out of-distribution detection: Solutions and future challenges,” arXiv preprint arXiv:2110.14051, 2021. https://arxiv.org/pdf/2110.14051.pdf
5. J. Hao, T. Ho Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. // Journal of Educational and Behavioral Statistics. Vol 44, Feb 2019. Doi: 10.3102/1076998619832248.
6. N. Alamdari, N. Kehtarnavaz. A Real-Time Smartphone App for Unsupervised Noise Classification in Realistic Audio Environments. // In Proc. IEEE Intl. Conf. on Consumer Electronics (ICCE), Jan 2019. 1-5. Doi: 10.1109/ICCE.2019.8662052.
7. D. Sculley. Web-scale k-means clustering. // In Proc. of 19th Intl. Conf. on World Wide Web, Apr. 2010. pp. 1177-1178. Doi: 10.1145/1772690.1772862.
8. J.L. Bentley. Multidimensional binary search trees used for associative searching. // Communications of the ACM. (1975) 18 (9): pp. 509–517. doi:10.1145/361002.361007.
9. D. Comaniciu, P. Meer. Mean shift: a robust approach toward feature space analysis // In Proc. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 603-619, May 2002, doi: 10.1109/34.1000236.

10. J.H. Ward, Jr. Hierarchical Grouping to Optimize an Objective Function, // In Journal of the American Statistical Association, 1963, vol 58, pp. 236–244.
11. M. Ester, H. P. Kriegel, J. Sander, X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise // In Proc. of the 2nd Inter. Conf. on Knowledge Discovery and Data Mining, 1996, pp. 226–231
12. M. Ankerst, M.M. Breunig, H.P. Kriegel, J. Sander. OPTICS: ordering points to identify the clustering structure. // In ACM Sigmod Record, 1999, vol. 28, No. 2, pp. 49-60.
13. T. Zhang, R. Ramakrishnan, M. Livny. BIRCH: An efficient data clustering method for large databases. // In ACM Sigmod Record, 1996, vol. 25, issue 2, pp. 103-114.

Одержано: 07.02.2024

Про авторів:

Рагозін Дмитро Васильович,
старший науковий співробітник.
Кількість публікацій в українських виданнях – більше 10.
Кількість зарубіжних публікацій – більше 5.
https://orcid.org/0000-0002-8445-9921

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії комп'ютерних обчислень,
професор кафедри інформаційних систем та технологій Національного технічного університету України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій в українських виданнях – понад 200.
Кількість наукових публікацій в зарубіжних виданнях – понад 90.
Індекс Хірша – 7.
http://orcid.org/0000-0002-8435-1451

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
E-mail: dmytro.rahozin@gmail.com,
dmytro.rahozin@ukr.net

УДК 51:681.3.0 http://doi.org/10.15407/pp2024.01.057

А.Ю. Дорошенко, Р.В. Кушніренко

АВТОМАТИЗАЦІЯ ГЛИБОКОГО НАВЧАННЯ НА ПРИКЛАДІ УТОЧНЕННЯ ЧИСЕЛЬНИХ МЕТЕОРОЛОГІЧНИХ ПРОГНОЗІВ

Зроблено короткий огляд застосування "глибокого навчання" до науково-технічних задач. Перевірено можливість застосування нейроеволюційного підходу до проєктування моделей "глибокого навчання", призначених для постпроцесингу результатів метеорологічного прогнозування (на прикладі приземної температури), отриманого за допомогою чисельних гідродинамічних методів. Показано, що в половині випадків і значення стандартного середнього-квадратичного відхилення (RMSE), і відсотка покращених прогнозів для нейроеволюційного підходу є кращими (а в окремих випадках набагато кращими) за відповідні значення для підібраної вручну архітектури.

Ключові слова: "глибоке навчання", автоматизація проєктування нейромереж, нейроеволюція, метеорологічне прогнозування.

Вступ

Велика кількість даних, що стала доступною протягом останнього десятиліття, спричинила революцію в дослідницьких і оперативних схемах геонаукової обробки, уможлививши використання "глибокого навчання" для задач, що стосуються атмосфери, поверхні суші та океану. Окрім збільшення доступності даних спостережень, цьому процесу сприяло, поміж іншого, підвищення швидкості їхньої передачі, що вже перевищує сотні терабайт на день [1]. Ці дані надходять від безлічі датчиків, зокрема, вони включають дані дистанційного зондування на висоті від кількох метрів до сотень кілометрів над Землею, а також спостереження на місці (на поверхні та під нею) за допомогою автономних датчиків.

Використання "глибокого навчання" для розв'язання проблем, що постають перед геонаукою, знаходиться ще в зародковому стані (на відміну від помітних успіхів у моделюванні впорядкованих послідовностей і даних із просторовим контекстом у сферах комп'ютерного зору, систем розпізнавання мови та керування [2], а також у таких наукових галузях як фізика [3], хімія [4] та біологія [5]). Хоча, як показують дослідження, його застосування в геонауці є перспективним, особливо в задачах класифікації, регресії, виявлення аномалій та прогнозування залежного від простору або часу стану. До прикладу, в дослідженнях [6,7] демонструється застосування "глибокого навчання" до проблеми прогнозування екстремальних погодних умов, задачі, проблемної для традиційного машинного навчання. Зазначимо, що особливо успішним є застосування архітектур "глибокого навчання" до виокремлення просторових і часових характеристик для визначення та класифікації екстремальних ситуацій (зокрема, штормів) у вихідних даних числової моделі прогнозування погоди. Важливим є те, що виявлення цих подій та створення прогнозів відбувається без використання суб'єктивних суджень людини або методів, які покладаються на заздалегідь визначені порогові значення для швидкості вітру та інших метеорологічних величин.

Підходи "глибокого навчання" класично поділяються на просторові (наприклад, згорткові нейронні мережі [8] для класифікації об'єктів) і послідовні (наприклад, рекурентні нейронні мережі для розпізнавання мовлення [9]). Однак останнім часом спостерігається все більша зацікавленість у поєднанні цих двох підходів. Прикладом цього поєднання є прогнозування відео та руху [10], проблема, яка має разючу подібність до багатьох динамічних геонаукових проблем. Уже існують дослідження, що починають застосовувати комбіновані згортково-рекурентні підходи до таких геонаукових проблем як прогнозування опадів [11]. Моделювання динаміки атмосфери та океану, моделювання поширення вогню чи руху ґрунту також є прикладами проблем, де

важлива просторово-часова динаміка, але наразі вони не отримали переваг від застосування комбінованих згортково-рекурентних підходів "глибокого навчання".

Коротко кажучи, подібність між типами даних, притаманних для класичних застосувань "глибокого навчання", і даних, з якими працює геонаука, є переконливим аргументом на користь проникнення "глибокого навчання" в геонауки. Зображення є аналогом двовимірних полів даних, що містять певні змінні за аналогією з триплетами кольорів (значення RGB) на фотографіях, тоді як відео можна пов'язати з послідовністю зображень, тобто з двовимірними полями, які змінюються у часі. Так само, природна мова та мовлення мають такі ж характерні особливості динамічних часових рядів, що їх мають дані, притаманні геонауковій сфері. Крім того, класифікація, регресія, виявлення аномалій і динамічне моделювання є типовими проблемами як для класичних застосувань "глибокого навчання", так і для геонаук.

Як бачимо, з розвитком "глибоких" нейронних мереж ми отримуємо все досконаліші архітектури, які можемо застосовувати у різних сферах, зокрема, в геонауці. Проте безплатний сир буває тільки в мишоловці, адже підвищення якості моделей є результатом усе більшої їх складності. Тому постає нова задача, а саме задача налаштування систем отримання рішень. Це налаштування полягає у підборі конфігураційних параметрів (гіперпараметрів). Якщо цих параметрів мало, то їх можна спробувати оптимізувати шляхом експериментів. Натомість сучасні "глибокі" нейронні мережі мають складну топологію та сотні гіперпараметрів. Окрім цього, як зазначалося вище, важить вибір архітектури нейромережевих моделей, оскільки успішність отриманого рішення часто залежить від цього вибору. Власне, останнім часом велика частка робіт у галузі "глибокого" навчання була присвячена мануальній розробці різних архітектур для розв'язання нових проблем [12,13].

Зазначимо, що проблема складності не є притаманною лише для нейронних мереж. Загалом розробка програмного забезпечення та багатьох інших інженерних систем стала надто складною для того, щоб люди могли її повністю оптимізувати. В результаті з'явився новий підхід до проєктування програмного забезпечення. Цей підхід залишає людям високорівневе проєктування, натомість з'ясування деталей лежить на оптимізаційних системах. Наприклад, люди проєктують програмну систему, а параметри та код низького рівня оптимізуються автоматично [14].

Подібний підхід можна застосувати до проєктування архітектур "глибоких" нейронних мереж. Для цього треба уміти давати раду трьом аспектам: проєктуванню компонентів архітектури, об'єднанню компонентів (створення топології мережі) та підбору гіперпараметрів (для кожного з компонентів та для системи в цілому). Звісно, що для кожного нового завдання матимемо окрему оптимізацію вищеперерахованого.

Це дослідження використовує підхід до автоматичного проєктування "глибоких" нейронних мереж, описаний у [15]. Своєю чергою, цей підхід використовує техніку нейроеволюції доповнених топологій [16]. Ця техніка поширюється на коеволюційну оптимізацію компонентів, топологій і гіперпараметрів.

Власне, дана стаття присвячена застосуванню нейроеволюційного підходу до проєктування моделей "глибокого навчання", призначених для постпроцесингу результатів прогнозу приземної температури, отриманого за допомогою чисельних гідродинамічних методів метеорологічного прогнозування. В статті також виконане порівняння якості прогнозу з результатами, отриманими раніше за допомогою ручного підбору нейромережевої архітектури [17,18]. Як буде видно, застосування нейроеволюційного підходу до задачі покращення чисельного метеорологічного прогнозування показує, що можна без особливих зусиль отримати результати кращі за отримані під час мануального проєктування архітектури нейромережевої моделі. Звісно, цей підхід є надзвичайно вимогливим до обчислювальної потужності, проте, враховуючи невеликий об'єм даних, він дає гарні результати.

Опис даних

"Глибоке навчання" як техніка виокремлення характерних особливостей даних суттєво залежить від якості, репрезентативності та цілісності використовуваних даних. Тому правильний відбір і підготовка даних є важливими факторами для отримання хороших узагальнюючих результатів.

Зокрема, відбір даних має бути спрямований на охоплення найбільш повної варіативності значень змінних, на яких базуватиметься власне навчання нейромережевої моделі. Хороші дані мають дозволяти моделі охоплювати зв'язки між змінними, на основі яких робиться прогноз. Водночас важливим є уникнення надлишковості у даних.

Нижче поданий опис даних, що були використані для дослідження, описаного у даній статті. Ці дані складалися з чотириелементних кортежів і містили наступну інформацію:

- дата,
- час за Гринвічем,
- прогнозоване значення температури (Fcst), за одну добу до моменту ініціалізації чисельної регіональної моделі,
- спостережуване значення температури (Obs).

Чисельною моделлю прогнозу погоди, результат роботи якої ми хочемо покращити, є модель однойменного європейського консорціуму COSMO (Consortium for Small-scale Modelling). Ця модель використовується в Українському гідрометеорологічному інституті ДСНС України та НАН України для наукових та прикладних задач, починаючи із липня 2011 р. [19]. Нагадаємо, що COSMO є негідростатичною моделлю, яка здатна ефективно відтворювати широкий спектр атмосферних процесів в масштабі мезо-β та мезо-γ. В основу динамічного ядра моделі покладено рівняння термо- та гідродинаміки, що описують потік у вологій атмосфері. Різноманітні фізичні процеси враховуються схемами параметризації [20].

Рис. 1 зображує розрахункову область чисельної регіональної моделі: кількість вузлів із заходу на схід – 209; кількість вузлів із півдня на північ – 101; кількість рівнів по вертикалі – 50; крок ~ 14 км.

Рис. 1. Розрахункова область моделі прогнозу погоди COSMO

Наявні дані охоплюють проміжок часу від 01.07.2012 до 31.03.2014, або 639 днів. Спостереження здійснювалися кожні три години, а саме о 00:00, 03:00, 06:00, 09:00, 12:00, 15:00, 18:00 і 21:00 за Гринвічем. Для цих же моментів часу обчислювався і прогноз регіональної моделі.

Таким чином, для кожної дати маємо по вісім кортежів. Відповідно, 639 днів дають 5112 кортежів.

Що ж до просторової приналежності, то дані охоплюють спостережувані значення і прогнози для станцій "Біла Церква", "Бориспіль", "Київ", "Миронівка", "Тетерів", "Фастів", "Чорнобиль" та "Яготин".

Нейроеволюційний підхід

Нагадаємо, що термін "нейроеволюція" позначає процес використання генетичних алгоритмів для підбору архітектур нейромережевих моделей.

У цьому дослідженні ми будемо використовувати одну з модифікацій нейроеволюції доповнених топологій. Даний підхід, відповідно до [16], ставить собі за мету боротися із трьома основними викликами, що постають перед рішеннями, які базуються на еволюції топології. Ці виклики та запропоновані рішення наведені нижче:

- Чи існує генетичне представлення, яке дозволяє розрізненим топологіям схрещуватись значущим чином? Рішення: використання історичних міток для виокремлення генів однакового походження.

- Як захистити від передчасного зникнення з популяції топологічну інновацію, для оптимізації якої потрібно кілька поколінь? Рішення: розподілення інновації між різними видами.
- Як протягом процесу еволюції мінімізувати топологію без залучення спеціально створеної для цього цільової функції, яка вимірює складність? Рішення: усе починається з мінімальної структури, яка еволюціонує лише за необхідності.

Зазначимо, що застосування запропонованих рішень до кожного з викликів є необхідним, оскільки ефективність нейроеволюційного процесу знижується із видаленням будь-якого з основних компонентів. З іншого боку, злагоджена робота цих складових створює новий перспективний підхід до розв'язання складних задач навчання з підкріпленням і не тільки.

Таким чином, архітектура нейроеволюції доповнених топологій була розроблена з урахуванням перерахованих вище проблем. Опис цього підходу подається відповідно до [16].

Дана архітектура репрезентує геном як список генів зв’язку, кожному з яких відповідають два вузлових гени. Кожний ген зв’язку містить інформацію про вхідний та вихідний вузли, ваговий коефіцієнт зв’язку, прапорець активності, а також історичну мітку для пошуку генів однакового походження під час кросинговеру.

Нейроеволюція доповнених топологій використовує мутації, що можуть змінювати і вагові коефіцієнти, і структуру мережі. Вагові коефіцієнти з’єднання змінюються як і в будь-якій іншій нейроеволюційній системі. Водночас з’єднання не зобов’язане змінюватись, воно може лишитись незмінним. Структурні мутації розширюють геном і можуть відбуватися двома шляхами: додається або нове з’єднання, або новий вузол. Коли відбувається мутація першого типу, у геном додається новий ген зв’язку, який з’єднує два раніше не з’єднані вузли. Другий тип мутації зумовлює розрив наявного з’єднання, на місці якого розміщується новий вузол. Старе з’єднання стає неактивним, натомість два нові з’єднання додаються до геному. Такий спосіб додавання вузлів уможливлює миттєву інтеграцію нових вузлів у мережу.

Мутації зумовлюють появу геномів різного розміру, іноді з абсолютно різними топологіями. Тут ми підходимо до першого виклику, означеного вище: знайти представлення геному, яке дозволятиме розрізненим топологіям схрещуватись значущим чином.

Схрещування (кросинговер) можливе лише тоді, коли система має можливість визначити, які гени є спільними для будь-яких двох особин в популяції. Ключове спостереження полягає в тому, що два гени з однаковим історичним походженням мають однакову структуру (хоча, можливо, із різними ваговими коефіцієнтами), оскільки вони обидва були отримані від одного гена-предка у певний момент у минулому. Отож, усе, що потрібно зробити системі, — це відстежувати історичне походження кожного гена.

На щастя, відстеження історичного походження потребує дуже мало обчислень. Щоразу, коли з’являється новий ген (через структурну мутацію), ми збільшуємо глобальний номер інновації та присвоюємо цьому гену. Таким чином, історичне походження кожного гена в системі відоме протягом еволюційного процесу, за це відповідають глобальні номери інновацій.

Історичні мітки дають системі, побудованій на основі нейроеволюції доповнених топологій, можливість визначати сумісні гени. Несумісні гени можуть бути або ексцесивними, або взагалі не перетинатися, залежно від того, чи їхні мітки лежать у межах, чи поза межами діапазону історичних міток кожного з батьків. У процесі кросинговеру в обох геномах впорядковуються гени з однаковими номерами інновації. Несумісні гени успадковуються від більш придатного батька або випадково, якщо кожний з батьків однаково придатний. Таким чином, історичні мітки дозволяють виконувати операцію кросинговеру без залучення дорогого топологічного аналізу.

Зазначимо, що описаний метод кросинговеру вирізняється своєю простотою і загальністю. Будь-які дві структури можуть бути об’єднані без необхідності жодного топологічного аналізу. Проблема поєднання то-

пологій перетворюється у проблему визначення відповідності історичних міток, а в такому вигляді її значно простіше розв'язувати.

Зазвичай зміна структури мережі призводить спочатку до зниження її ефективності. Тому необхідним є механізм захисту топологічних (структурних) інновацій від передчасного зникнення з популяції. Нейроеволюція доповнених топологій має такий механізм. Це досягається особливим чином: спочатку індивіди конкурують у межах власного виду, а не в рамках цілої популяції. Так, топологічні інновації захищені від передчасного зникнення та мають час для оптимізації своєї структури, перш ніж доведеться конкурувати з іншими видами в популяції.

Історичні мітки дають змогу системі розділити популяцію на види за топологічною подібністю. Кількість ексцесивних генів і генів, що взагалі не перетинаються, є природними параметрами для вимірювання міри близькості геномів. Чим меншу спільну еволюційну історію мають два геноми, тим більше вони розрізнені, а отже, менш сумісні. Тож, ми можемо виміряти міру сумісності різних структур як зважену лінійну комбінацію кількості ексцесивних генів та генів, що не перетинаються.

Це метричне співвідношення дозволяє нам ввести порогове значення для сумісності двох геномів. Ми тестуємо геноми один за одним. Якщо відстань від геному до навмання обраного представника виду менша за порогове значення, даний геном вважається приналежним до цього виду. При цьому кожний геном ми вважаємо приналежним до першого виду, де ця умова виконується, що гарантує те, що жоден геном не міститься більше, ніж в одному виді.

Декілька слів про третій виклик, що постає перед нейроеволюційними підходами: мінімізація топології отриманого рішення. Рішення, що його пропонує нейроеволюція доповнених топологій, — просте. Ми починаємо з найпростішої структури та ускладнюємо її лише за необхідності. Нова структура вводиться поступово в міру того, як відбуваються структурні мутації. В результаті виживають лише ті структури, які виявляються ефективними у сенсі значень цільової функції. Зазначимо, що такий підхід можливий завдяки описаному вище механізму захисту інновацій. Без цього механізму нові структури не могли б виживати, оскільки, як зазначено вище, щойно утворена структура спочатку призводить до зниження ефективності.

Опис експерименту

Як було зазначено вище, основною ідеєю даного дослідження є перевірка можливості застосування нейроеволюційного підходу до проєктування моделей "глибокого навчання", призначених для постпроцесингу результатів прогнозу приземної температури, отриманого за допомогою чисельних гідродинамічних методів метеорологічного прогнозування.

Для кожної метеорологічної станції за допомогою нейроеволюції доповнених топологій тренувалася власна нейромережева модель, яка мала б якнайкраще виокремити фізичні особливості конкретного пункту спостереження. Таким чином, ми маємо 8 натренованих моделей.

Що до даних, то для кожної метеорологічної станції уся їх сукупність була розбита на три класи: тренувальні (період з 01.07.2012 до 30.06.2013; 365 днів), валідаційні (період з 01.07.2013 до 31.10.2013; 123 дні) і тестувальні (з 01.11.2013 до 01.04.2014; 151 день).

Зауважимо, що цей поділ на три класи (набори) — це загальна практика для "глибокого навчання" [21]. Тренувальний набір є найбільшим і використовується для оновлення вагових коефіцієнтів моделі шляхом зворотного розповсюдження помилки або інших алгоритмів навчання. Другий набір, валідаційний, використовується виключно для налаштування гіперпараметрів: кількості шарів, типів шарів, функцій активації, цільових функцій, швидкості навчання тощо. Ключовою метою цього налаштування є підвищення здатності мережі до узагальнення, щоб гарантувати, що мережа добре функціонуватиме на невідомих для неї даних. Третій набір даних — це тестовий набір, раніше невідомі дані, які використовуються для оцінювання мережі після налаштування.

Зазначимо, що усі експерименти проводилися з використанням відкритого нейромережевого інтерфейсу Keras [22], відкри-

тої програмної бібліотеки для "глибокого навчання" TensorFlow [23] та фреймворку для нейроеволюції TFNE (Tensorflow-Neuroevolution Framework) [24]. Програмний код був написаний мовою Python [25].

Отримані результати

Ефективність запропонованого підходу оцінювалась за двома критеріями. Перший з них — це корінь середнього квадратичного відхилення (RMSE), стандартний спосіб оцінювання похибок у метеорологічній науці. Інша метрика — відсоток випадків, в яких прогноз покращено (або принаймні не погіршено) порівняно з чисельним. Як бачимо з таблиці, в більшості випадків і значення RMSE, і відсоток покращених прогнозів для нейроеволюційного підходу є кращими (не гіршими) за відповідні значення для підібраної вручну архітектури.

Таблиця. Порівняльна таблиця

Станція	RMSE (NN), °C	Покращення, %	RMSE (NE), °C	Покращення, %
Біла Церква	1.89	54.81	1.96	54.81
Бориспіль	2.00	53.21	1.70	54.56
Київ	1.98	53.97	1.85	56.84
Миронівка	2.01	57.48	1.84	58.42
Тетерів	2.02	52.29	2.03	50.79
Фастів	1.86	54.68	1.86	54.42
Чорнобиль	2.01	50.59	1.94	52.53
Яготин	2.12	52.21	2.05	49.40

Висновки

На прикладі прогнозів моделі COSMO приземної температури повітря для восьми метеорологічних станцій Київської області та відповідних їм даних фактичних спостережень було перевірено можливість застосування нейроеволюційного підходу до проєктування моделей "глибокого навчання", призначених для постпроцесингу результатів прогнозу приземної температури, отриманого за допомогою чисельних гідродинамічних методів метеорологічного прогнозування.

Було показано, що в половині випадків і значення кореня середнього квадратичного відхилення, і відсотка поліпшених прогнозів для нейроеволюційного підходу є кращими (а в окремих випадках набагато кращими) за відповідні значення для підібраної вручну архітектури.

Отримані результати дають підстави для застосування нейроеволюційного підходу до коригування прогнозів інших неперервних метеорологічних величин.

References

1. Agapiou, A., 2017. Remote sensing heritage in a petabyte-scale: satellite data and heritage Earth Engine© applications. *International Journal of Digital Earth*, *10*(1), pp.85-102.
2. LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *nature*, *521*(7553), pp.436-444.
3. Bhimji, W., Farrell, S.A., Kurth, T., Paganini, M., Prabhat and Racah, E., 2018, September. Deep neural networks for physics analysis on low-level whole-detector data at the LHC. In *Journal of Physics: Conference Series* (Vol. 1085, p. 042034). IOP Publishing.
4. Schütt, K.T., Arbabzadah, F., Chmiela, S., Müller, K.R. and Tkatchenko, A., 2017. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, *8*(1), p.13890.
5. Alipanahi, B., Delong, A., Weirauch, M.T. and Frey, B.J., 2015. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology*, *33*(8), pp.831-838.
6. Liu, Y., Racah, E., Correa, J., Khosrowshahi, A., Lavers, D., Kunkel, K., Wehner, M. and Collins, W., 2016. Application of deep convolutional neural networks for detecting extreme weather in climate datasets. *arXiv preprint arXiv:1605.01156*.
7. Racah, E., Beckham, C., Maharaj, T., Ebrahimi Kahou, S., Prabhat, M. and Pal, C., 2017. Extremeweather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. *Advances in neural information processing systems*, *30*.
8. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning

applied to document recognition. *Proceedings of the IEEE*, *86*(11), pp.2278-2324.
9. Sak, H., Senior, A. and Beaufays, F., 2014. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.
10. Oh, J., Guo, X., Lee, H., Lewis, R.L. and Singh, S., 2015. Action-conditional video prediction using deep networks in atari games. *Advances in neural information processing systems*, *28*.
11. Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K. and Woo, W.C., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, *28*.
12. Che, Z., Purushotham, S., Cho, K., Sontag, D. and Liu, Y., 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, *8*(1), p.6085.
13. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).
14. Hoos, H.H., 2012. Programming by optimization. *Communications of the ACM*, *55*(2), pp.70-80.
15. Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N. and Hodjat, B., 2024. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing* (pp. 269-287). Academic Press.
16. Stanley, K.O. and Miikkulainen, R., 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation*, *10*(2), pp.99-127.
17. Doroshenko, A.Y., Shpyg, V.M. and Kushnirenko, R.V., 2023. Deeplearning-based approach to improving numerical weather forecasts. *PROBLEMS IN PROGRAMMING*, (3), pp.91-98.
18. Doroshenko, A.Y. and Kushnirenko, R.V., 2023. Recurrent neural networks for the problem of improving numerical meteorological forecasts. *PROBLEMS IN PROGRAMMING*, (4), pp.90-97.
19. Shpyg, V., Budak, I., Pishniak, D. and Poperechnyi, P., 2013, November. The application of regional NWP models to operational weather forecasting in Ukraine. In *CAS Technical Conference (TECO) on" Responding to the Environmental Stressors of the 21st Century" Available from: http://www. wmo. int/pages/prog/arep/ cas/documents/Ukraine-NWPModels. pdf [Accessed 27/02/2020]*.
20. Doms, G. and Baldauf, M., 2011. A description of the nonhydrostatic regional COSMO-Model Part I: dynamics and numerics. *Deutscher Wetterdienst, Offenbach*.
21. Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep learning*. MIT press.
22. https://keras.io/
23. https://www.tensorflow.org/
24. https://tfne.readthedocs.io/en/latest/
25. https://www.python.org/

Одержано: 03.03.2024

Про авторів:

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу ІПС НАНУ та
професор кафедри інформаційних систем
та технологій КПІ імені Ігоря Сікорського.
Кількість наукових публікацій
в українських виданнях – понад 200.
Кількість наукових публікацій
в зарубіжних виданнях – понад 90.
Індекс Гірша — 7
http://orcid.org/0000-0002-8435-1451,

Кушніренко Роман Владиславович,
аспірант.
Кількість наукових публікацій
в українських виданнях – 5.
https://orcid.org/0000-0002-1990-8727.

Місце роботи авторів:

Інститут програмних систем НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (38)(044) 526-60-33.
E-mail:
doroshenkoanatoliy2@gmail.com,
roman.kushnirenk@gmail.com.

УДК 004.41 http://doi.org/10.15407/pp2024.01.064

А. М. Глибовець, Т. А. Чернова, М. М. Глибовець

РОЗРОБКА МЕТОДОЛОГІЇ ІМПЛЕМЕНТАЦІЇ ТРАНЗАКЦІЙ В РОЗПОДІЛЕНИХ СИСТЕМАХ З МІКРОСЕРВІСНОЮ АРХІТЕКТУРОЮ

У роботі описано аналіз проблематики використання мікросервісної архітектури в розподілених системах. Наголос зроблено на гнучкості у виборі технологій, масштабованості та організації команд, які працюють над заданими мікросервісами, технічних і доменних проблемах реалізації транзакцій у порівнянні з монолітною системою. Основну увагу приділено транзакціям, оскільки вони забезпечують дотримання атомарності, консистентності, ізольованості та стійкості над декількома сервісами.

У процесі аналізу сучасних підходів та рішень для роботи з транзакціями в розподілених системах було виявлено, що одним з ефективних рішень є використання патерну Transactional Outbox. Представлено його реалізацію у вигляді Spring starter. Останній додається до системи, конфігурується та полегшує використання транзакцій і публікацію подій, які є частинами транзакції у мікросервісній архітектурі.

Ключові слова: розподілена система, розподілені транзакції, мікросервісна архітектура, патерн Transactional Outbox, асинхронне спілкування, Kafka, Debezium.

Вступ

Наразі для розробки складних програмних систем дедалі частіше використовується мікросервісна архітектура (МА). Це рішення дозволяє розбивати систему на невеликі незалежні компоненти (застосунки) та отримувати швидку розгортку, масштабованість та гнучкість. Однак в таких розподілених системах (РС) виникають певні труднощі керування транзакціями, оскільки вони вимагають забезпечення атомарності, консистентності, ізольованості та стійкості над декількома сервісами [1-3].

Транзакції у базі даних – робоча операційна одиниця [2] для роботи із базою даних, певна послідовність змін, що виконуються в логічному порядку користувачем або програмою, яка працює із базою даних. Якщо відбувається будь-яка операція з інтерфейсом користувача, у базі даних виконується транзакція. Основні концепції транзакцій описуються абревіатурою ACID - Atomicity, Consistency, Isolation, Durability (Атомарність, Узгодженість, Ізольованість, Довговічність).

Атомарність гарантує, що будь-яка транзакція буде зафіксована тільки цілком і кожен крок транзакції виконається повністю. Якщо ж одна з операцій в послідовності завершиться із помилкою, то вся транзакція буде скасована. Існує поняття «відкату змін» (rollback), за якого всі зміни у зворотному порядку скасовуватимуться. У разі виникнення помилки під час проходження транзакції, користувач не побачить ніяких змін.

Узгодженість (консистентність) забезпечує те, що будь-яка завершена транзакція фіксує тільки допустимі результати. Під час виконання принципу узгодженості, база даних повинна завжди переходити із одного несуперечливого стану в інший несуперечливий стан. Умова узгодженості є необхідною для підтримки четвертої властивості транзакції - довговічності.

Ізольованість визначає те, що результат кожної транзакції не повинен залежати від виконання інших паралельних транзакцій. На практиці повна ізольованість важко досяжна. Тому вводиться поняття «рівні ізольованості». Послаблення ізольованості транзакцій призводить до відомих проблем: «брудне читання» (dirty read), «втрачене оновлення» (lost update), «брудний запис» (dirty write), «невідтворюване читання» (nonrepeatable read), «фантомне читання» (phantom read), «перекошене читання або запис» (read and write skew) [2]. У системах із МА, де застосовуються транзакції, розподілені між кількома сервісами, проблема ізольованості постає особливо гостро.

РС - це сукупність незалежних між собою компонентів, які взаємодіють та координують свої дії для досягнення спільної мети [1]. Ці компоненти можуть бути апаратними або програмними, можуть знаходитись на різних комп'ютерах та бути пов'язані спільною мережею. Компоненти взаємодіють між собою, передаючи повідомлення, використовуючи віддалені виклики процедур або інші форми комунікації. Цей принцип взаємодії дозволяє кожному компоненту працювати незалежно і спільно. Управління такою системою є складним процесом, оскільки вимагає вирішення таких проблем, як багатопоточність, консистентність даних, стійкість до помилок, збереження цілісності даних та масштабованості системи.

У МА застосунок – це набір невеликих, незалежних сервісів, кожен з яких має певну бізнес-функціональність. Ці сервіси можуть бути розроблені, розгорнуті та масштабовані незалежно один від одного, що дозволяє досягти більшої гнучкості та динамічності порівняно з традиційними монолітними архітектурними рішеннями [2].

Спілкування між сервісами в МА відбувається за використанням чітко визначених API, застосовуючи протоколи комунікації, такі як REST, черги повідомлень. Це дозволяє системі легше розвиватися та масштабуватися, оскільки зміни в одному сервісі не обов'язково потребують змін у всій системі. Однак архітектура мікросервісів також вносить власний набір викликів, включаючи управління консистентністю даних, балансування навантаження та моніторинг, трейсинг та логування [4]. Ці виклики потрібно ретельно досліджувати для забезпечення стійкості та масштабованості системи.

Тому виникає необхідність ефективних рішень для роботи з транзакціями в РС. Виникли спеціалізовані патерни та методології, які пропонують використання окремих підходів до реалізації такої роботи. Наприклад, патерн "Transactional Outbox (TO)" розроблений для забезпечення атомарності операцій та публікації подій як частини транзакційного процесу.

Метою цієї статті є опис аналізу існуючих загальних рішень імплементації транзакцій в РС мікросервісного типу та створеної методології реалізації розподілених транзакцій на базі черг повідомлень. Було виявлено, що одним з ефективних рішень є використання патерну ТО. Представлено розроблену нами його реалізацію у вигляді Spring starter. Останній додається до системи, конфігурується та полегшує використання транзакцій і публікацію подій, які є частинами транзакції в МА.

1. Проблеми в мікросервісній архітектурі

Під час проєктування комунікації між мікросервісами потрібно враховувати: налаштування правильної комунікації між сервісами, можливі затримки в мережі, проблеми з доступністю серверів та затримки між запитами [5]. У разі налаштування комунікації між мікросервісами важливою складовою є забезпечення відмовостійкості. Потрібно враховувати і консистентність даних. До прикладу, атомарне оновлення даних (або всі операції успішні, або жодна не успішна) є не простим викликом у побудові МА.

Управління консистентністю даних включає розв'язання двох основних проблем: роботу з багатопоточністю та відмовостійкістю. В багатопоточному середовищі, коли кілька процесів або потоків намагаються одночасно отримувати доступ та змінювати ті самі дані, може виникати неконсистентність даних, оскільки різні вузли можуть мати різні варіації тих самих даних. Відмова ж може відбуватися, коли вузол або мережеве з'єднання зазнають пошкоджень. Наприклад, сервіс в поточний момент часу не відповідає або повертає помилку, що призводить до часткових або неконсистентних оновлень даних.

Існує кілька підходів до управління консистентністю даних. Один із підходів - використання розподілених транзакцій (РТ), які забезпечують виконання групи операцій на кількох вузлах атомарно. РТ - це транзакції, що охоплюють кілька баз даних або систем, які можуть знаходитися на різних серверах або навіть у різних географічних місцях.

У РТ координатор транзакції відповідає за координацію дій усіх вузлів системи. Координатор спочатку запускає транзакцію, а потім надсилає запити до учасників систем для виконання потрібних операцій. Якщо всі вузли успішно виконують свої операції, координатор підтверджує транзакцію. Однак, якщо будь-яка з систем не виконує свою операцію, то координатор скасовує всю транзакцію, скасовуючи водночас усі зміни, зроблені учасниками систем.

Хоча розподілені транзакції можуть бути корисні для керування узгодженістю даних у МА, вони також можуть створювати власний набір проблем та труднощів, таких як додаткове навантаження на продуктивність та операційну складність.

2. Патерни, що полегшують роботу з розподіленими транзакціями

Розглянемо деякі відомі мікросервісні патерни, що вирішують проблеми, пов’язані з РТ та їх реалізації [6-7].

Шаблон двофазного коміту (2PC) є класичним розподіленим транзакційним шаблоном, який використовується для координації транзакцій між декількома системами. У цьому патерні існує координатор, що ініціює транзакцію, й інші учасники (мікросервіси, вузли системи), що отримують повідомлення від координатора та зберігають транзакцію [8]. Координатор визначає готовність кожного вузла зберегти транзакцію і, у випадку готовності усіх сервісів, відправляє друге повідомлення, аби вказати всім сервісам зберегти транзакцію. Якщо якийсь сервіс не готовий зберегти транзакцію, координатор відправляє повідомлення про відмову від транзакції. Цей патерн вирішує проблему консистентності даних в РС забезпечуючи те, що всі сервіси здійснюють або скасовують транзакцію у координований спосіб (за допомогою сервісу координатора). Він уможливлює забезпечення атомарності та консистентності між кількома системами, але може спричиняти проблеми, пов'язані з масштабованістю, доступністю та продуктивністю.

Серед відомих реалізацій цього патерну можна виділити наступні: Atomikos (система керування транзакціями, заснована на Java), Bitronix (менеджер транзакцій, заснований на Java), NServiceBus (платформа для передачі повідомлень та інтеграції на основі .NET) .

Шаблон Saga є альтернативою шаблону 2PC. Принцип роботи шаблону Saga полягає у тому, що після виконання локальної транзакції база даних оновлюється, а повідомлення про подію публікується для ініціювання наступної локальної транзакції до баз даних. У Saga використовуються такі терміни, як: compensating transaction (транзакція, що скасовує зміни, виконані попередньою транзакцією), countermeasure (дизайн техніка, що використовується для компенсації нестачі рівня ізоляції), compensatable transaction (транзакція, яку можна компенсувати), pivot transaction (транзакція, що є маркером для саги: якщо транзакція успішна, то сага продовжуватиме ряд дій для завершення послідовності транзакцій) та retriable transaction (повторна транзакція). Цей шаблон дозволяє забезпечити правильність виконання розподіленої між мікросервісами транзакції, а кожну локальну транзакцію можна компенсувати за допомогою її компенсуючої транзакції. Saga також гарантує, що всі операції будуть завершені успішно або відповідні компенсаційні транзакції будуть запущені для скасування раніше виконаної роботи. Є два види координації транзакцій у Saga: хореографія та оркестрація.

Заслуговують на увагу такі реалізації цього патерну: Uber Cadence (РС оркестрування робочих процесів, яка містить підтримку як саг на основі хореографії, так і саг на основі оркестрації), AxonIQ (відкрита платформа для створення мікросервісів, заснованих на подіях, яка включає підтримку саг), Lightbend Akka (набір інструментів та середовище виконання для побудови високопродуктивних, розподілених та відмовостійких систем, яка містить підтримку саг).

Окремо стоїть модель Eventual consistency. Це спосіб управління консистентністю даних в РС без використання традиційних транзакцій. У моделі дозволяється сервісам незалежно оновлювати свої локальні сховища даних, а послідовна консистентність досягається за допомогою

асинхронних оновлень та вирішення конфліктів. Хоча цей підхід може бути більш масштабованим та гнучким, ніж традиційні транзакції, він потребує обережного керування конфліктами даних та може призводити до тимчасових неузгодженостей.

Компенсаційний патерн - це спосіб скасувати наслідки невдалої транзакції без потреби повного відкату системи. У цьому патерні виконується компенсуюча транзакція для скасування наслідків початкової транзакції. Цей патерн часто використовується в поєднанні з патерном саги для керування локальними транзакціями в межах сервісу.

Патерн TO надає спосіб забезпечення консистентності даних у PC, коли сервіс повинен публікувати події як частину транзакції. Це легкий та ефективний спосіб керування координацією подій між декількома системами, не вводячи складності традиційних механізмів PT. Він дозволяє сервісу публікувати події у спосіб, який консистентний з локальною базою даних, не потребуючи механізмів 2PC або інших складних механізмів, як от saga, чи компенсаційний патерн. Це легкий і ефективний спосіб керування координацією подій між декількома системами.

Патерн часто реалізують на базі фреймворку Spring Cloud Stream. Він призначений для створення мікросервісних застосунків, які працюють на основі подій. Він дозволяє публікувати та отримувати події за допомогою різних систем повідомлень, таких як Apache Kafka, RabbitMQ та Google Pub/Sub. Його модель програмування для реалізації патерну TO базується на API Spring Transaction.

Говорячи про відомі реалізацій даних мікросервісних патернів варто зазначити, що їх можна реалізовувати різноманітними способами, підлаштовуючись під клієнтські запити та технічну базу застосунку. Наприклад, Eventuate.io є платформою для створення та управління мікросервісами, що працюють за принципом event-driven applications. Вона включає й підтримку шаблону TO, надає набір клієнтських бібліотек для Java та Spring, які спрощують інтеграцію шаблону TO в застосунки Spring Boot.

3. Transactional Outbox як асинхронний спосіб вирішення проблеми обміну даними в розподілених системах

Під час використання MA поширеним є послуговування своїми локальними сховищами даних - database per service.

Наприклад, сервіс, що відповідає за створення замовлень, зберігає замовлення в реляційній базі даних. Водночас сервіс може бажати надіслати подію про нове замовлення до Apache Kafka, щоб поширити цю інформацію на інші зацікавлені сервіси. Виконання цих двох дій може призвести до неузгодженості даних: нове замовлення буде збережене в локальній базі даних, але відповідного повідомлення до Kafka не буде надіслано (наприклад, через проблему з мережею). Або навпаки, ми можемо надіслати повідомлення до Kafka, але не зможемо зберегти замовлення в локальній базі даних. Обидві ситуації є небажаними та можуть призвести до того, що нібито успішно створене замовлення не буде відправлене. Або буде створений запит на відправку, але в самому сервісі замовлень не буде відомостей про відповідне замовлення.

Оптимальним рішенням цієї проблеми була б зміна лише одного ресурсу: або ж збереження даних на сервері, або ж відправка повідомлення і в подальшому оновлювати інший ресурс на основі успішності першого. Якщо розглянути спершу зміну ресурсу Apache Kafka, то послідовність подій виглядатиме наступним чином: сервіс, що відповідає за збереження замовлень, не буде зберігати замовлення в базі, натомість відправить подію в брокер повідомлень та підпишеться на цей же топік для отримання результату про успішну або неуспішну відправку. Отож, змінюється лише один ресурс за раз, і, якщо щось піде не так, ми одразу дізнаємося про це та повідомимо замовнику, що запит не вдалося виконати. Оскільки сервіс підписується на топік в Kafka, він буде отримувати сповіщення про доставку нового повідомлення в цей же топік і зможе зберегти нове замовлення на закупівлю у своїй базі даних.

Чим погане таке рішення? Якщо в системі виникатиме потреба додати функ-

ціонал на зчитування усіх замовлень в базі, то наш сервіс не зможе читати свої власні записи (read your own write semantic). Припустимо, що сервіс замовлень також має API для пошуку всіх замовлень певного клієнта. У разі виклику цього API безпосередньо після розміщення нового замовлення через асинхронну обробку повідомлень з топіка Kafka може статися так, що замовлення на закупівлю ще не було збережено в базі даних сервісу і, отже, його не буде повернуто цим топіком. Це може призвести до плутанини. Наприклад, користувачі можуть пропустити новостворені замовлення в своїй історії покупок та загалом втратити вкрай важливу інформацію про замовлення.

Існують способи впоратися з цією ситуацією. Так, сервіс може зберігати нові замовлення на закупівлю в оперативній пам'яті та відповідати на наступні запити на основі цих даних. Однак це швидко стає непростою задачею у разі реалізації складніших запитів або врахуванні того, що сервіс замовлень може складатися з кількох вузлів у кластерному середовищі, що потребуватиме передачі цих даних в межах кластера. Більш слушним рішенням буде зміна спершу стану бази, а потім відправка повідомлення, базуючись на зміні цього ресурсу. Дане рішення якраз таки можна імплементувати, використовуючи патерн ТО [9].

Патерн ТО вирішує завдання збереження консистентності та надійності даних у розподілених транзакційних середовищах шляхом збереження даних та подій, пов’язаних з ними в межах однієї бази даних, та гарантує публікацію збережених подій. Публіковані події відображають зміни стану системи, які потрібно повідомити іншим вузлам розподіленої системи. Додаючи події в таблицю "outbox", патерн забезпечує їх надійне зберігання в межах тієї самої атомарної транзакції разом із основною операцією бізнес логіки. Цей підхід надає гарантії консистентності даних, оскільки або всі зміни комітяться разом, або жодна з них не виконується (із скасуванням однієї скасовуються усі). Існує й окремий фоновий процес, відомий як "паблішер" подій або диспетчер подій. Він читає події з таблиці "outbox" та публікує їх у відповідний посередник повідомлень або систему обміну повідомленнями – це може бути Apache Kafka, RabbitMQ та інші. Шляхом розподілення процесу публікації подій від основної транзакції, шаблон ТО допомагає покращити продуктивність, масштабованість та стійкість системи.

Один із базових підходів реалізації патерну ТО є розробка та використання власних сервісів, що будуть зчитувати події з таблиці Outbox, менеджити їхній подальший життєвий цикл та керувати обробкою різноманітних помилкових сценаріїв. Основна перевага такого підходу є його гнучкість, оскільки він дозволяє налаштувати логіку обробки повідомлень відповідно до конкретних вимог.

У патерні кожен мікросервіс підтримує таблицю "outbox" у власній локальній базі даних. Ця таблиця є своєрідним буфером, де зберігаються події та повідомлення до їх поширення у зовнішні системи (наприклад, до публікування у меседж брокер).

Структура таблиці outbox table зазвичай залежить від конкретної реалізації та вимог МА. Однак є кілька загальних атрибутів, які часто присутні в таблиці вихідної скриньки, такі як message_id, message_payload, status, timestamp та інші. Конкретна структура та додаткові поля можуть варіюватися залежно від деталей реалізації та вимог архітектури мікросервісів.

У такій реалізації патерну існують спеціальні консьюмери, які відповідають за періодичні запити до таблиці "outbox" для отримання не переданих та не зчитаних раніше повідомлень. Цими консьюмерами можуть бути окремі мікросервіси або окремі модулі власного застосунку. Після отримання повідомлення з таблиці "outbox" спеціальний консьюмер обробляє його відповідно до визначеної бізнес-логіки. Цей процес може включати форматування повідомлення, виклик API або сервісів та обробку можливих помилок чи повторних спроб. У розробці такого консьюмера варто зважати на необхідність надання гарантованого читання, відправки та підтвердження оброблених повідомлень. Підтвердження читання, до прикладу, можна реалізовувати оновлюючи статус у таблиці "outbox" .

Підхід із використанням спеціальних консьюмерів надає деталізований контроль і налаштування споживання та обробки повідомлень. Він дозволяє адаптувати логіку обробки під задоволення конкретних вимог, таких як перетворення даних, перевірка безпеки або правил валідації.

Хоча даний варіант реалізації патерну сприяє гнучкості і можливості налаштування, він також має кілька потенційних недоліків, які варто врахувати. Серед вагомих недоліків можна вважати збільшений обсяг розробки та потреби підтримки в експлуатації. Необхідно проєктувати, реалізовувати, тестувати та підтримувати компоненти власних консьюмерів. А це збільшує загальний обсяг роботи з розробки та потребує більше зусиль для їхньої підтримки. Масштабування власних споживачів може також викликати певні складнощі, особливо у роботі з великою кількістю повідомлень та подій або одночасним доступом до таблиці outbox. Потрібно використовувати відповідні механізми балансування навантаження та дбати про ефективну та масштабовану обробку повідомлень.

4. Поняття WAL та CDC

Change Data Capture (CDC) та Write-Ahead Log (WAL) є пов'язаними концепціями в контексті систем баз даних, але вони служать різним цілям.

"Write-Ahead Log" (WAL) є технікою, яка часто використовується в системах баз даних для забезпечення стійкості даних та підтримки консистентності транзакцій. Вона передбачає запис змін, пов'язаних із транзакціями, у журнальний файл перед їхнім застосуванням до фактичних файлів даних бази даних. WAL виступає як послідовний журнал усіх змін, зроблених у базі даних, як успішно здійснених, так і незавершених транзакцій [10].

Коли транзакція змінює дані в базі даних, відповідні зміни спочатку записуються в WAL. Це гарантує безпечне зберігання журналу на надійному носії, перш ніж модифікації застосовуються до самої бази даних. Завдяки цьому підходу система бази даних гарантує, що в разі збою або аварії системи зміни, записані в WAL, можуть бути відтворені та застосовані для відновлення бази даних у консистентний стан.

WAL надає кілька переваг. Вона покращує продуктивність бази даних, дозволяючи буферизувати модифікації в пам'яті та записувати їх на диск у ефективнішому послідовному порядку. Вона також забезпечує цілісність даних, надаючи надійний запис транзакцій, що дозволяє відновлення після збоїв та скасування операцій. Крім того, WAL дозволяє реплікацію бази даних та реалізацію резервних серверів, реплікуючи журнал на інші системи для потреб синхронізації.

Загалом використання WAL підвищує стійкість, надійність та можливість відновлення систем баз даних, роблячи його фундаментальною технікою для забезпечення консистентності та доступності даних.

Change Data Capture (CDC) - це техніка, що використовується у галузі інтеграції та синхронізації даних для захоплення та передачі змін даних у реальному часі з баз даних до цільових систем [11]. CDC дозволяє постійно відстежувати та витягувати зміни даних, такі як вставки, оновлення та видалення, у міру їх виникнення в джерелі бази даних. Ці зловлені зміни даних потім перетворюються у формат, який може бути використаний іншими застосунками або реплікований до інших баз даних. CDC дозволяє ефективну та майже в реальному часі реплікацію даних, синхронізацію та інтеграцію між різноманітними системами, забезпечуючи консистентність та актуальність даних у екосистемі. Це широко використовується в сценаріях, де своєчасна та точна передача даних є критичною.

Зв'язок між CDC та WAL полягає в тому, що CDC часто використовує інформацію, записану в WAL, для захоплення та відстеження змін. Оскільки WAL містить послідовний журнал усіх модифікацій, зроблених у базі даних, його можна використовувати як надійне джерело для видобутку окремих змін та їх передачі іншим системам або споживачам.

Аналізуючи WAL, механізми CDC можуть ідентифікувати конкретні зміни, що сталися, включаючи редаговані рядки, стовпці та значення. Потім вони можуть зафіксувати ці зміни та перетворити їх у фор-

мат, придатний для передачі, як наприклад, створення подій зміни або оновлення реплік бази даних.

5. Debezium як спосіб імплементації транзакцій

Debezium - це розподілена платформа з відкритим кодом, яка використовує техніку Capture Data Capture (CDC) для отримання та передачі змін даних у реальному часі із журналів транзакцій бази даних (використовуючи WAL) [12]. Вона інтегрується з різноманітними популярними базами даних, такими як MySQL, PostgreSQL, MongoDB та інші за рахунок конекторів до баз даних. А також надає незамінні функції, такі, як здатність захоплювати зміни в усіх аспектах обробки даних включно із вставками, оновленням та видаленням. Під час налаштування для роботи з певною базою даних Debezium підключається до журналу транзакцій бази даних, який, як правило, реалізується за допомогою WAL. WAL реєструє всі модифікації, зроблені в базі даних, включаючи вставки, оновлення та видалення, в послідовному та стійкому журнальному файлі. Debezium зчитує зміни, записані в WAL, та перетворює їх у потік подій змін. Ці події змін потім перетворюються в стандартизований формат, такий як JSON або Avro, і стають доступними для споживання застосунками або системами вниз по ланцюжку.

Завдяки використанню підходу CDC на основі WAL, Debezium забезпечує надійний та мінімальний вплив захоплення даних, не навантажуючи додатково джерело бази даних [13]. Така ефективність доповнюється легко розширюваною архітектурою, яка може пристосовуватися до ситуацій стійкості до відмов, сприяючи простій синхронізації та інтеграції з численними платформами. Тому внесок Debezium у створення процесів стрімінгу в реальному часі, разом із наданням масштабованих реактивних потоків для сучасних розподілених фреймворків, неможливо недооцінити. Цей підхід мінімізує вплив на продуктивність джерела бази даних та забезпечує майже реальний час та точну реплікацію та інтеграцію даних між різними системами.

Debezium побудований на основі Apache Kafka. На відміну від підходу, заснованого на пулінгу даних, Debezium отримує події з дуже низьким навантаженням на систему та практично в режимі реального часу [12]. Debezium поставляється з CDC-конекторами для кількох баз даних, таких як MySQL, Postgres та SQL Server і після отримання даних може передавати їх іншим сервісам, зокрема, може публікувати подію в меседж брокер. Надійність та запобігання втрати даних є важливими характеристиками платформи.

Debezium надає пріоритет надійності, використовуючи журнали транзакцій, такі як Write-Ahead Log, що надаються системами баз даних, з якими він інтегрується. Ці журнали є надійним та міцним джерелом для отримання інформації про зміни даних. Debezium підключається до журналу транзакцій і зчитує записані в ньому зміни, гарантуючи, що жодні зміни не пропускаються або не втрачаються. Під час захоплення змін з журналу транзакцій Debezium запам’ятовує свою останню оброблену подію, щоб у разі перезапуску або відмови системи, Debezium міг продовжити читати події з того місця, де він зупинився, і продовжити читання змін без дублювання, тобто не читати повторно одні й ті ж дані. Цей механізм забезпечує цілісність даних та запобігає втраті даних у разі будь-яких збоїв або відмов.

Apache Kafka, брокер повідомлень, який використовується разом з Debezium, також відіграє важливу роль у забезпеченні надійності та стійкості даних. Він реплікує дані на кількох брокерах у кластері, забезпечуючи їхню обробку в разі відмов. Кожне повідомлення або подія, яка публікується в Kafka, зберігається у сховищі, яке називається "топіками". Топіки розбиваються на частини, і кожна частина реплікується на кількох брокерах. Цей механізм реплікації забезпечує ситуацію, коли навіть у разі відмови деяких брокерів або вузлів, дані все ще доступні і можуть споживатися користувачами.

Debezium використовує ці та інші можливості надійності Kafka, публікуючи отримані з таблиці outbox зміни подій в топіки Kafka. Після публікації Kafka відповідає за забезпечення їхньої міцності та дос-

тупності. Події можуть бути споживані консьюмерами або оброблені іншими застосунками.

6. Spring Boot Starter та його роль в реалізації транзакцій

Spring Boot Starter - є ключовою складовою частиною фреймворку Spring Boot, що надає спрощений спосіб налаштування та запуску застосунків через об'єднання залежностей та налаштувань для конкретних функціональностей або компонентів. Зазвичай Spring Boot Starter за замовчуванням включає підібраний набір бібліотек, класів автоконфігурації та налаштувань, необхідних для включення певної функції або інтеграції з певною технологією [14].

За допомогою Spring Boot Starter розробники можуть швидко додавати необхідні можливості та функціонал до своїх застосунків, не потребуючи ручного налаштування залежностей або написання стандартного коду. Ці стартери упаковують необхідні залежності та конфігурацію, що дозволяє розробникам зосередитися на написанні бізнес-логіки, а не гаяти час на складнощі налаштування та інтеграції різних компонентів.

Spring Boot Starter надає широкий спектр функціоналу включно із підключенням до баз даних, веб-розробкою, безпекою, обміном повідомленнями, кешуванням тощо. Кожен стартер дотримується певного підходу і надає послідовний спосіб інтеграції пов'язаних технологій у застосунки Spring Boot. На прикладі стартеру "spring-boot-starter-web" можна побачити залежності та конфігурації, необхідні для розробки веб-застосунків з використанням Spring MVC, який включає в себе цей стартер. При доданні spring-boot-starter-web у проєкт, автоматично імпортується кілька інших залежностей, таких як:

- Spring MVC: основний компонент фреймворку Spring для створення веб-застосунків, що надає архітектуру MVC та обробляє відображення маршрутів запитів, обробку запитів та генерацію відповідей.
- Embedded Servlet Container: вбудований контейнер сервлетів (наприклад, Tomcat, Jetty або Undertow), що дозволяє запускати веб-застосунки без потреби в зовнішній конфігурації сервера.
- Spring Web: модуль, що надає додаткові функції та утиліти для роботи з вебом, включно із обробкою HTTP-запитів та відповідей, обробкою форм, зв'язуванням даних, валідацією та обробкою помилок.
- Jackson JSON: бібліотека включена для підтримки серіалізації та десеріалізації даних у форматі JSON (JavaScript Object Notation). Вона дозволяє легко конвертувати об'єкти Java в JSON та навпаки.

Крім наданих стартерів, розробники також можуть створювати власні стартери для упакування уже використовуваних компонентів або бібліотек, специфічних для їхніх застосунків чи домену. Це сприяє модульності коду, повторному використанню та стандартизації в проєктах організації.

7. Методологія імплементації транзакцій

Проаналізувавши проблематику транзакцій в РС, ми розробили власну методологію імплементації транзакцій на базі патерну ТО та черг повідомлень, яка спрямована на їх вирішення з наголосом на забезпеченні послідовності, консистентності та надійності даних у множинних сервісах. На рисунку 1 наведена структура запропонованої методології на прикладі розробленої системи для предметної області – замовлень товарів.

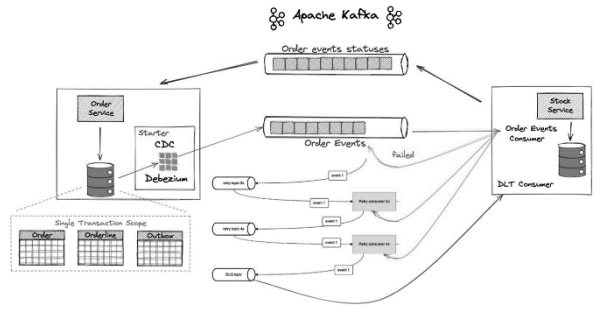

Рис. 1. Методологія імплементації транзакцій

Нехай система замовлень реалізована у вигляді двох мікросервісів: Order Service, що отримує замовлення товарів, та Stock Service, який на базі цих замовлень здійснює певні перевірки товару та інші ма-

ніпуляції для успішного завершення життєвого циклу замовлення. Обидва мікросервіси реалізовані на Java. Вони використовують Spring Boot Framework, JPA/Hibernate для доступу до своїх баз даних, які реалізовані на MySQL. Проте для нашої методології немає чіткої прив'язки до бази даних чи то фреймфорку, оскільки вона дозволяє реалізацію з використанням різноманітних технологій, зокрема, таких як: CDI, PostgreSQL, WildFly та інших.

Сервіс замовлень надає простий REST API для створення замовлень. Сервіс складу за допомогою Apache Kafka отримує події, експортовані сервісом замовлень та створює відповідні записи у власній базі даних MySQL.

Для експортування подій у Apache Kafka сервіс замовлень використовує розроблений Spring Boot Starter, що дбає про налаштування таблиці outbox. Запис подій до таблиці відбувається в межах тієї ж транзакції, що і збереження замовлення в базі даних. Стартер самостійно дбає про моніторинг та зчитування даних таблиці outbox, використовуючи WAL, CDC, Debezium, та публікацію зчитаних даних до Apache Kafka.

Після зчитування подій з Apache Kafka, сток сервіс сповіщає про вдале або провальне зчитування та обробку даних, використовуючи Apache Kafka. Ці події пізніше сервіс замовлень читає та змінює статус замовлення з pending на closed / canceled. Також цей сервіс дбає про відмовостійкість завдяки повторному читанню даних, гарантує обробку даних та відповідь у вигляді публікації статусу обробки у відповідний топік в Apache Kafka.

Дотримуючись цієї методології, можна забезпечити надійний обмін даними між мікросервісами, обробку операцій у межах транзакції та відповідних подій, відправлених іншим сервісам, атомарно та послідовно. Навіть у разі виникнення проблеми під час надсилання подій, їх можна спробувати повторно відправити, або відтворити з таблиці outbox, забезпечуючи потрібну послідовність в РС. Цим можна скористатися і для збереження цілісності та послідовності даних, одночасно дозволяючи асинхронну комунікацію між сервісами.

8. Практична реалізація

На основі запропонованої методології був розроблений Spring boot Starter, який полегшує конфігурацію роботи з транзакціями в РС, що публікують події як частину транзакції. Він має вигляд спеціальної бібліотеки для полегшення впровадження патерну Transactional Outbox в застосунок, побудований на основі Java. Основною метою стартера є спрощення процесу збереження та публікації подій у РС.

Стартер включає необхідні залежності та конфігурації, а також надає зручні методи для збереження подій у відведеній таблиці (outbox) і передачі їх у Debezium для подальшого розповсюдження. Використовуючи цей стартер, розробники можуть ефективно впроваджувати патерн TO, покладаючись на потужність Debezium для зчитування та розповсюдження подій у РС, а також на гарантовану доставку та цілісність даних за рахунок використання Apache Kafka як базової технологій для Debezium.

Інтеграція застосунку зі стартером розпочинається з його підключення у вигляді залежності (Рис. 2.):

```
<dependency>
    <groupId>com.chernova</groupId>
    <artifactId>transactional-outbox-starter</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

Рис. 2. Підключення стартеру до клієнтського застосунку

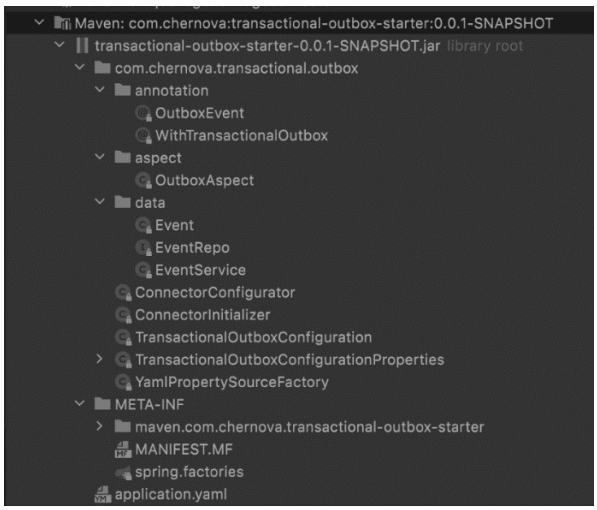

Рис. 3. – Підключений стартер

Рисунок 3 відображає доданий стартер до застосунку. Для коректної роботи стартеру та підтримки роботи з outbox table та Apache Kafka, необхідно передати наступні конфігурації, що будуть зчитані стартером під час ініціалізації проєкту. Connector.class - вказує клас конектора, який використовується для підключення до джерела даних (наприклад, значення конектор класу io.debezium.connector.mysql.MySqlConnector, означає підключення до MySQL бази даних). database.hostname - вказує ім'я хоста, на якому розташована база даних клієнта. atabase.port – визначає порт, на якому доступна база даних клієнта. database.user - уточнює ім'я користувача для підключення до бази даних клієнта. database.password - вказує пароль для підключення до бази даних клієнта. database.server.id - задає унікальний ідентифікатор сервера, який використовується для розпізнавання транзакцій. database.server.name - визначає унікальне ім'я сервера, яке застосовується для ідентифікації каналу змін даних у Debezium. database.history.kafka.bootstrap.servers - вказує адресу та порт сервера Apache Kafka, який використовується для зберігання історії змін бази даних (до прикладу “broker:9092”). database.history.kafka.topic - уточнює тему Kafka, де зберігається історія змін бази даних. key.converter.schema.registry.url: вказує адресу та порт сервера реєстрації схем Avro.

```yaml
transactionaloutbox:
  enabled: true
  connector:
    class: io.debezium.connector.mysql.MySqlConnector
  database:
    hostname: mysql
    port: 3306
    user: debezium
    password: dbz
    server:
      id: 42
      name: asgard
    history:
      kafka:
        topic: dbhistory.demo
        bootstrap:
          servers: broker:9092
    key:
      converter:
        schema:
          registry:
            url: http://schema-registry:8081
    value:
      converter:
        schema:
          registry:
            url: http://schema-registry:8081
```

Рис. 4. Приклад конфігурації стартера в yaml файлі

value.converter.schema.registry.url - визначає адресу та порт сервера реєстрації схем Avro.

Ці конфігурації мають бути надані у yaml файлі й використовуються для налаштування Debezium з метою підключення до бази даних MySQL, зчитування змін у реальному часі та перетворення їх на події, які можна розповсюджувати через Apache Kafka. Приклад конфігурації наведено на рисунку 4.

Для керування підключенням чи відключенням роботи стартера під час запуску застосунку необхідно також задати значення true/false для конфігурації transactionaloutbox.enabled.

Для збереження подій до outbox table, зчитування подій з outbox table та публікацію їх до Apache Kafka, необхідно додати до методу сервісу анотацію @WithTransactionalOutbox (рис.5).

```java
@Transactional
@WithTransactionalOutbox
public PurchaseOrder addOrder(@OutboxEvent PurchaseOrder order) {
    return purchaseOrderRepo.save(order);
}
```

Рис. 5. Використання анотації WithTransactionalOutbox доданого стартера

В цьому прикладі стартер в межах однієї транзакції зі збереженням замовлення до бази збереже подію до таблиці outbox. Для реалізації такого функціоналу стартер використовує можливості Spring, а саме Spring AOP (рис.6).

```java
@Around(value = "@annotation(withTransactionalOutboxAnnotation)")
public Object withTransactionalOutbox(ProceedingJoinPoint joinPoint, WithTransactionalOutbox withTransactionalOutboxAnnotation)
        throws Throwable {
    Object result = joinPoint.proceed();
    String jsonData = "{}";

    MethodSignature signature = (MethodSignature) joinPoint.getSignature();
    Annotation[][] annotations = signature.getMethod().getParameterAnnotations();
    Object[] args = joinPoint.getArgs();

    Object annotatedArgument;
    for (int i = 0; i < args.length; i++) {
        for (Annotation annotation : annotations[i]) {
            if (annotation instanceof OutboxEvent) {
                annotatedArgument = args[i];
                jsonData = objectMapper.writeValueAsString(annotatedArgument);
                break;
            }
        }
    }
    ...

    return result;
}
```

Рис. 6. Spring AOP для реалізації логіки стартера

Застосувавши усі необхідні налаштування стартера, можна інтегрувати засто-

сунок із патерном ТО і таким чином забезпечити атомарність операцій та консистентність даних між різними сервісами. Більше того, додаючи стартер до свого застосунку, можна отримати ряд можливостей, які надаються технологіями Debezium та Apache Kafka.

9. Рекомендації щодо налаштування системи на базі розробленої методології

Під час проектування та реалізації РС, які використовують методологію імплементації транзакцій, варто врахувати кілька рекомендацій.

Насамперед, важливо переконатися, що компоненти системи мають слабку зв'язність та спілкуються за допомогою асинхронного обміну повідомленнями для розділення транзакційної обробки від зовнішніх сервісів. Такий підхід спілкування сприяє стійкості та масштабованості [15-16]. Не менш важливим є використання надійного брокеру повідомлень для комунікації між мікросервісами та гарантованої надійної доставки повідомлень. У ролі такого брокеру можуть бути Kafka або RabbitMQ.

Однією з важливих компонент у розробці дистрибутивної системи є впровадження механізмів обробки помилок та повторного читання даних у разі помилок, управління відмовами та забезпечення консистентності даних. Із цією метою можна використовувати механізми Retryable Topic та Dead Letter Topic [17-18].

Також важливо належну увагу приділити надійності консьюмерів [19]. Оскільки консьюмери виконують обробку повідомлень із черги та виконують певні операції або змінюють стан системи на основі цих повідомлень, важливо забезпечити, щоб ці операції гарантували безпеку та надійність системи, приводили обробку даних до однакового результату незалежно від кількості повторних повідомлень. Це важливо для забезпечення консистентності даних та уникнення небажаних побічних ефектів. Наприклад, якщо консьюмер створює запис у базі даних на основі отриманого повідомлення, то необхідно гарантувати, щоб у базі даних було створено тільки один унікальний запис, навіть якщо повідомлення буде оброблено кілька разів. Також важливим у проєктуванні консьюмерів є використання унікальних ідентифікаторів для оброблених повідомлень або збереження стану процесу обробки, що забезпечить ідемподентну обробку.

10. Тестування

Основними аспектами тестування була перевірка вдалої інтеграції Debezium з існуючими компонентами та технологіями в системі. Це включило перевірку забезпечення правильної конфігурації та налаштування стартера, який базується на імплементації ТО патерну. Також було оцінено інтеграцію з брокером повідомлень Kafka та перевірено публікацію повідомлень до брокеру, наявність повторного читання даних за необхідності.

На нашому прототипі були здійснені перевірки поведінки системи у випадку наступних сценаріїв виникнення помилки:

- у разі спроби збереження замовлення до клієнтської бази та перевірки системи на скасування публікації повідомлення;
- у разі публікації повідомлення до брокера повідомлень та перевірки системи на скасування збереження даних про замовлення до клієнтської бази;
- у разі повідомлення консьюмером у сток сервісі та перевірки системи на повторне читання з використанням retryable механізму;
- після кількох невданих спроб обробки повідомлень, коли консьюмер в сток сервісі не може обробити повідомлення та перевірити систему на відправку повідомлення до DLT топіку;
- після успішної обробки повідомлення сток сервіс відправив статус успішної обробки до відповідного топіку і сервіс замовлень, базуючись на даній відповіді, змінив статус замовлення на завершений;
- після неуспішної обробки повідомлення сток сервіс відправив статус проблеми обробки до відповідного топіку і сервіс замовлень, базуючись на даній відповіді, змінив статус замовлення на скасований.

Як результат оцінювання система коректно відпрацювала в обох успішних та помилкових сценаріях. Варто зазначити, що даний прототип має правильно налаштовані механізми повторної спроби читання та обробки даних, оскільки у випадку виникнення помилки, спроєктований механізм повторної спроби автоматично повторював операцію, для подолання тимчасових проблем, таких як проблеми з мережею або тимчасова недоступність ресурсів. Повторні спроби обробки виконувались із використанням стратегій публікацій подій до DLT топіку, щоб уникнути перевантаження системи та забезпечити затримку між послідовними спробами.

Висновки

У процесі аналізу сучасних підходів та рішень для роботи з транзакціями в РСах було виявлено, що одним із ефективних рішень є використання патерну Transactional outbox. Він дозволяє зберігати події, пов'язані з транзакціями, в окремій таблиці, що уможливлює їхню подальшу інтеграцію з іншими сервісами через застосунки або механізми, як-от, через брокер повідомлень Apache Kafka.

Проведений аналіз дозволив розробити методологію імплементації транзакцій в РС, а також створити стартер для полегшення такої роботи з розподіленими транзакціями та публікацією подій, що є частинами транзакції в чергу повідомлень. Цей стартер надає зручний і стандартизований спосіб інтеграції патерну TO в Spring Boot застосунки, забезпечуючи автоматичне керування транзакціями та публікацію подій.

Оцінювання прототипу показало ефективність впровадження даного підходу в РС. Розроблена методологія та стартер можуть допомогти забезпечити надійну обробку розподілених транзакцій та публікацію подій.

Література

1. Maarten van Steen, Andrew S. Tanenbaum. "A brief introduction to distributed systems." Web. 2016. https://www.distributed-systems.net/my-data/papers/2016.computing.pdf
2. Fowler M. Microservices [Електронний ресурс] / M. Fowler, J. Lewis. – 2014. – Режим доступу до ресурсу: https://martinfowler.com/articles/microservices.html.
3. Яшина О.М., Кравчук О.А. Дослідження мікросервісної архітектури, архітектурний стиль REST та їх сучасна реалізація на Java. Вісник ХНУ: технічні науки, Номер: №5, 2020, с. 106-114.
4. Trzaska, Mariusz. “Technical challenges of creating an IT system in microservices architectural style using cloud services” Web. 2022 https://users.pja.edu.pl/~mtrzaska/Files/PraceMagisterskie/220217-Grabowski.pdf
5. Newman S. Monolith To Microservices / Sam Newman., 2019. – 301 с. – (2).
6. Fowler, Martin. “How to break a Monolith into Microservices” Web. 2018 https://martinfowler.com/articles/break-monolith-into-microservices.html
7. Richardson C. Microservices Patterns With examples in Java / Chris Richardson., 2018. – 520 с.
8. Fowler, Martin “Two Phase Commit” Web. https://martinfowler.com/articles/patterns-of-distributed-systems/two-phase-commit.html
9. Transactional Outbox Pattern https://www.linkedin.com/pulse/transactional-outbox-pattern-distributed-pratik-pandey (дата звернення 08.07.23)
10. Chola, Abhinal “Understanding Write Ahead Logging: 4 Comprehensive Aspects” Web. https://hevodata.com/learn/write-ahead-logging/
11. Spring Blog. “Case Study: Change Data Capture (CDC) Analysis with CDC Debezium source and Analytics sink in Real-Time” Web. https://spring.io/blog/2020/12/14/case-study-change-data-capture-cdc-analysis-with-cdc-debezium-source-and-analytics-sink-in-real-time
12. Debezium Documentation. Web. https://debezium.io/documentation/reference/stable/index.html
13. Hevodata. “3 Easy Steps to Decode CDC Using Debezium Spring Boot” Web. https://hevodata.com/learn/debezium-spring-boot/
14. Spring Boot Starters documentation. Web https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#using.build-systems.starters

15. Rajabi M. How to Avoid Coupling in Microservices Design [Електронний ресурс] / Mariam Rajabi. – 2020. – URL: https://www.capitalone.com/tech/softwareengineering/how-to-avoid-loose-coupled-microservices/. (дата звернення 15.09.21)
16. Walpita P. Coupling and Cohesion in Microservices [Електронний ресурс] / Priyal Walpita. – 2020. – URL: https://priyalwalpita.medium.com/coupling-and-cohesion-inmicroservices-235ed9203843. (дата звернення 18.09.21)
17. Baeldung Blog "Implementing Retry in Kafka Consumer" Web. https://www.baeldung.com/spring-retry-kafka-consumer
18. Medium. "Dead Letter Queue (DLQ) in Kafka" Web. https://towardsdatascience.com/dead-letter-queue-dlq-in-kafka-29418e0ec6cf
19. Richardson, Chris. "Pattern: Idempotent Consumer" Web. https://microservices.io/patterns/communication-style/idempotent-consumer.html

Одержано: 01.02.2024

Про авторів:

Глибовець Андрій Миколайович,
доктор технічних наук, професор,
декан факультету інформатики
Національного університету
«Києво-Могилянська академія».
Кількість наукових публікацій
в українських виданнях – понад 40.
Кількість наукових публікацій
в зарубіжних виданнях – 8.
Індекс Хірша – 3.
https://orcid.org/0000-0003-4282-481X,

Глибовець Микола Миколайович,
доктор фізико-математичних наук,
професор факультету інформатики
Національного університету
«Києво-Могилянська академія».
Кількість наукових публікацій
в українських виданнях – понад 200.
Кількість наукових публікацій
в зарубіжних виданнях – більше 10.
Індекс Хірша – 12, https://orcid.org/0000-0002-3853-2171

Чернова Тетяна Андріївна,
магістр факультету інформатики
Національного університету
«Києво-Могилянська академія»

Місце роботи авторів:

Національний університет
«Києво-Могилянська академія»,
04070, м. Київ, вул. Сковороди 2.
E-mail: a.glybovets@ukma.edu.ua,
glib@ukma.edu.ua, t.chernova@ukma.edu.ua

УДК001.2 http://doi.org/10.15407/pp2024.01.077

С. Д. Штовба, М. В. Петричко

ІДЕНТИФІКАЦІЯ РІВНЯ СПОРІДНЕНОСТІ НАУКОВИХ СПЕЦІАЛЬНОСТЕЙ НА ОСНОВІ ДАНИХ СИСТЕМИ DIMENSIONS

Ідентифіковано рівні спорідненості наукових спеціальностей у межах Австралійсько–Новозеландської стандартної класифікації наук ANZCRC-2020. Ідентифікація здійснена з використанням інформаційної системи Dimensions шляхом аналізу 33.8 млн публікацій за 2019–2023 рр. Рівень спорідненості оцінено за індексом Жаккара. Встановлено, що із 14535 можливих пар спеціальностей, лише 131 пара має значущу спорідненість, за індексом Жаккара, що перевищує 0.05. З них для 20 пар спеціальностей рівень спорідненості є високим, а для 61 пари – середнім.

Ключові слова: ідентифікація, класифікація наук, спорідненість спеціальностей, аналіз даних, індекс Жаккара, наукові публікації, підбір рецензентів, наукометрія, Dimensions, ANZSRC-2020.

Вступ

Управління науковою діяльністю здійснюється в рамках деякої системи класифікації наук. В Україні одночасно діє дві системи класифікації наук: трирівнева та дворівнева. Трирівнева система формалізована «Переліком наукових спеціальностей», який складається з 27 галузей та 488 спеціальностей. Галузь утворюють від 3 до 133 спеціальностей. В деяких галузях спеціальності об'єднані у групи, наприклад, в галузі «Технічні науки» утворено 20 груп спеціальностей. З 2015 р. діє і дворівнева система класифікації освітньо-наукової діяльності, яка формалізована «Переліком галузей знань і спеціальностей, за якими здійснюється підготовка здобувачів вищої освіти». Відповідно до його поточної версії є 28 галузей знань, кожна з яких містить від 1 до 9 спеціальностей. Загальна кількість спеціальностей дорівнює 121. Є багато інших систем класифікації наук, як національного рівня, так і міжнародного. Поміж національних систем виділимо трирівневу Австралійсько–Новозеландську класифікацію ANZCRC-2020 [1], яка включає 22 галузі та 171 спеціальність та класифікацію Організації економічного співробітництва та розвитку [2], яка включає 6 галузей та 42 спеціальності.

Вдала система класифікації наук дозволяє краще зрозуміти особливості філософських процесів пізнання у різних науках, дослідити історичний розвиток та взаємодію різних галузей знань, підвищити ефективність пошуку документів і наукової інформації та удосконалити адміністрування та управління дослідженнями [3].

Метою статі є ідентифікація поточного рівня спорідненості наукових спеціальностей. Оцінки рівня спорідненості спеціальностей необхідні для ухвалення рішень під час вирішення таких задач як:

- удосконалення системи класифікації наук, як-от, об'єднання спеціальностей або зміна їхньої галузевої належності;
- виявлення схожих наукових та освітніх установ для налагодження нових кооперативних зв'язків або проведення їх реорганізації;
- автоматизація підбору рецензентів для експертизи дисертацій, рукописів статей, заявок на гранти тощо;
- формування міждисциплінарних наукових досліджень та освітніх програм.

Огляд літератури та ідея дослідження

Найчастіше ідентифікація спорідненості наукових спеціальностей здійснюється шляхом аналізу цитування [4]. Ідея оцінювання спорідненості за цитованістю полягає в тому, що спорідненість між спеціальностями збільшується, якщо в статті за однією спеціальністю є посилання на статтю з іншої спеціальності. Також використовуються і методи статистичного аналізу тексту, зокрема, в [5] запропоновано

лінгвістичний підхід для дослідження організації та еволюції наукових галузей у Web of Science. За лінгвістичним підходом спорідненість визначається частотами появи слів у контексті конкретних спеціальностей. У роботі [5] порівнюються 3 підходи до визначення спорідненості спеціальностей: на основі експертної класифікації; на основі цитувань; на основі лінгвістичної схожості. Спорідненість оцінюється за метрикою розбіжності (*dissimilarity*) – чим менше значення метрики, тим більша спорідненість і навпаки. Експерименти здійснено для статей з Web of Science, що розмічені за трирівневою системою класифікації наук. Виявлено, що підходи на основі цитувань та лінгвістичного аналізу дають подібні значення спорідненості, водночас вони значно відрізняються від експертних оцінок. Експертна оцінка дає ідеалізоване уявлення про спорідненість спеціальностей, в той час як аналіз цитувань дозволяє виявити соціальну спорідненість спеціальностей, а лінгвістичний підхід дозволяє виявити змістовну (когнітивну) спорідненість. В [6] рівень спорідненості ідентифікують за кількістю цитувань зі статті певного дослідника на статті у журналах з Web of Science, де кожен із журналів віднесено до однієї із предметних областей Web of Science. В [7] ідентифікація здійснюється за допомогою показників різноманітності за розподілом посилань між спеціальностями з урахуванням відстані між ними. Деякі дослідження [8, 9] використовують одночасно аналіз цитувань та лінгвістичний аналіз для виявлення спорідненості. Дещо менш поширеним підходом є ідентифікація за аналізом графу колаборацій, який описує належність співавторів до наукових спеціальностей [10]. Як математична модель використовується індекс різноманітності Стірлінга.

Усі вищезгадані підходи вимагають опрацювання великих інформаційних ресурсів. Окрім того, методи, що базуються на аналізі цитування, є досить інерційними. Неможливо миттєво оцінити міждисциплінарність нової роботи, оскільки для її цитування потрібен певний час. На противагу згаданим підходам, у статті [11] запропоновано швидкий метод оцінювання спорідненості наукових спеціальностей відповідно до системи класифікаці ANZSRC-2008. Метод оснований на текстовому аналізі з використанням сервісів інформаційної системи Dimensions. Розрахунок спорідненості спеціальностей здійснюється за індексом Жаккара як відношення кількості спільних публікацій двох спеціальності до загальної кількості публікацій за цими двома спеціальностями. Нещодавно Dimensions перейшла на оновлену систему класифікації ANZSRC-2020 [1]. Також база публікацій значно оновилася. Відповідно отримані в [11] оцінки рівня спорідненості втратили актуальність. Тому нижче здійснюється ідентифікація спорідненості наукових спеціальностей за методом [11] на новій джерельній базі і за новою системою класифікації наук. Ідентифікація здійснюється за публікаціями за період 2019–2023 рр.

Початкові дані для ідентифікації спорідненості спеціальностей

ANZSRC-2020 - це трирівнева класифікація наук за схемою: галузі (Divisions), спеціальності (Groups) та області (Fields). Уся наука розділена на такі 22 галузі:

30 Agricultural, Veterinary and Food Sciences;
31 Biological Sciences;
32 Biomedical and Clinical Sciences;
33 Built Environment and Design;
35 Commerce, Management, Tourism and Services;
34 Chemical Sciences;
36 Creative Arts and Writing;
37 Earth Sciences;
38 Economics;
39 Education;
40 Engineering;
41 Environmental Sciences;
47 Language, Communication and Culture;
48 Law and Legal Studies;
42 Health Sciences;
43 History, Heritage and Archaeology;
44 Human Society;
46 Information and Computing Sciences;
49 Mathematical Sciences;
50 Philosophy and Religious Studies;

51 Physical Sciences;
52 Psychology.

Кожна галузь об'єднує від 3 до 19 спеціальностей. Назва спеціальності складається з цифрового коду та змістовної частини. Код складається з номера галузі та порядкового номера спеціальності в межах галузі. Всього є 171 спеціальність, список наведено нижче:

3001 Agricultural Biotechnology;
3002 Agriculture, Land and Farm Management;
3003 Animal Production;
3004 Crop and Pasture Production;
3005 Fisheries Sciences;
3006 Food Sciences;
3007 Forestry Sciences;
3008 Horticultural Production;
3009 Veterinary Sciences;
3101 Biochemistry and Cell Biology;
3102 Bioinformatics and Computational Biology;
3103 Ecology;
3104 Evolutionary Biology;
3105 Genetics;
3106 Industrial Biotechnology;
3107 Microbiology;
3108 Plant Biology;
3109 Zoology;
3201 Cardiovascular Medicine and Haematology;
3202 Clinical Sciences;
3203 Dentistry;
3204 Immunology;
3205 Medical Biochemistry and Metabolomics;
3206 Medical Biotechnology;
3207 Medical Microbiology;
3208 Medical Physiology;
3209 Neurosciences;
3210 Nutrition and Dietetics;
3211 Oncology and Carcinogenesis;
3212 Ophthalmology and Optometry;
3213 Pediatrics;
3214 Pharmacology and Pharmaceutical Sciences;
3215 Reproductive Medicine;
3301 Architecture;
3302 Building;
3303 Design;
3304 Urban and Regional Planning;
3401 Analytical Chemistry;
3402 Inorganic Chemistry;
3403 Macromolecular and Materials Chemistry;
3404 Medicinal and Biomolecular Chemistry;
3405 Organic Chemistry;
3406 Physical Chemistry;
3407 Theoretical and Computational Chemistry;
3501 Accounting, Auditing and Accountability;
3502 Banking, Finance and Investment;
3503 Business Systems in Context;
3504 Commercial Services;
3505 Human Resources and Industrial Relations;
3506 Marketing;
3507 Strategy, Management and Organisational Behaviour;
3508 Tourism;
3509 Transportation, Logistics and Supply Chains;
3601 Art History, Theory and Criticism;
3602 Creative and Professional Writing;
3603 Music;
3604 Performing Arts;
3605 Screen and Digital Media;
3606 Visual Arts;
3701 Atmospheric Sciences;
3702 Climate Change Science;
3703 Geochemistry;
3704 Geoinformatics;
3705 Geology;
3706 Geophysics;
3707 Hydrology;
3708 Oceanography;
3709 Physical Geography and Environmental Geoscience;
3801 Applied Economics;
3802 Econometrics;
3803 Economic Theory;
3901 Curriculum and Pedagogy;
3902 Education Policy, Sociology and Philosophy;
3903 Education Systems;
3904 Specialist Studies in Education;
4001 Aerospace Engineering;
4002 Automotive Engineering;
4003 Biomedical Engineering;
4004 Chemical Engineering;
4005 Civil Engineering;
4006 Communications Engineering;

4007 Control Engineering, Mechatronics and Robotics;
4008 Electrical Engineering;
4009 Electronics, Sensors and Digital Hardware;
4010 Engineering Practice and Education;
4011 Environmental Engineering;
4012 Fluid Mechanics and Thermal Engineering;
4013 Geomatic Engineering;
4014 Manufacturing Engineering;
4015 Maritime Engineering;
4016 Materials Engineering;
4017 Mechanical Engineering;
4018 Nanotechnology;
4019 Resources Engineering and Extractive Metallurgy;
4101 Climate Change Impacts and Adaptation;
4102 Ecological Applications;
4103 Environmental Biotechnology;
4104 Environmental Management;
4105 Pollution and Contamination;
4106 Soil Sciences;
4201 Allied Health and Rehabilitation Science;
4202 Epidemiology;
4203 Health Services and Systems;
4204 Midwifery;
4205 Nursing;
4206 Public Health;
4207 Sports Science and Exercise;
4208 Traditional, Complementary and Integrative Medicine;
4301 Archaeology;
4302 Heritage, Archive and Museum Studies;
4303 Historical Studies;
4401 Anthropology;
4402 Criminology;
4403 Demography;
4404 Development Studies;
4405 Gender Studies;
4406 Human Geography;
4407 Policy and Administration;
4408 Political Science;
4409 Social Work;
4410 Sociology;
4601 Applied Computing;
4602 Artificial Intelligence;
4603 Computer Vision and Multimedia Computation;
4604 Cybersecurity and Privacy;
4605 Data Management and Data Science;
4606 Distributed Computing and Systems Software;
4607 Graphics, Augmented Reality and Games;
4608 Human-Centred Computing;
4609 Information Systems;
4610 Library and Information Studies;
4611 Machine Learning;
4612 Software Engineering;
4613 Theory of Computation;
4701 Communication and Media Studies;
4702 Cultural Studies;
4703 Language Studies;
4704 Linguistics;
4705 Literary Studies;
4801 Commercial Law;
4802 Environmental and Resources Law;
4803 International and Comparative Law;
4804 Law in Context;
4805 Legal Systems;
4806 Private Law and Civil Obligations;
4807 Public Law;
4901 Applied Mathematics;
4902 Mathematical Physics;
4903 Numerical and Computational Mathematics;
4904 Pure Mathematics;
4905 Statistics;
5001 Applied Ethics;
5002 History and Philosophy of Specific Fields;
5003 Philosophy;
5004 Religious Studies;
5005 Theology;
5101 Astronomical Sciences;
5102 Atomic, Molecular and Optical Physics;
5103 Classical Physics;
5104 Condensed Matter Physics;
5105 Medical and Biological Physics;
5106 Nuclear and Plasma Physics;
5107 Particle and High Energy Physics;
5108 Quantum Physics;
5109 Space Sciences;
5110 Synchrotrons and Accelerators;
5201 Applied and Developmental Psychology;
5202 Biological Psychology;
5203 Clinical and Health Psychology;
5204 Cognitive and Computational Psychology;
5205 Social and Personality Psychology.

На сьогодні в системі Dimensions проіндексовано понад 140 млн наукових публікацій. Біля 80% від усіх публікацій категоризовано за спеціальностями, тобто віднесено до однієї чи кількох спеціальностей [1]. Категоризація публікацій у Dimensions здійснена на основі машинного навчання, переважно за аналізом змісту назв, анотацій та ключових слів. За останні 5 років у системі Dimensions проіндексовано 33.8 млн публікацій, які є джерельною базою дослідження. Публікації за спеціальностями розподілені нерівномірно (рис. 1). Децильний коефіцієнт дорівнює 14.9. Найбільше публікацій – 3188112 віднесено до спеціальності 3202, найменше – 3339 – до спеціальності 3606. Навіть найменш популярна спеціальність має достатню кількість публікацій, щоб на їх основі отримати достовірні статистичні висновки.

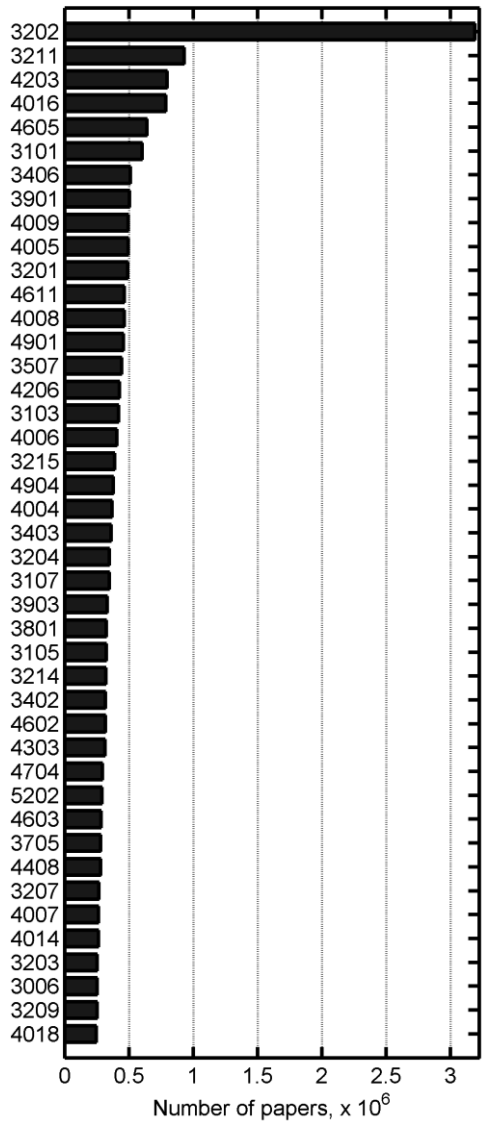

Рис. 1. Перший квартиль розподілу публікацій за спеціальностями за 2019–2023 рр.

Ідентифікація спорідненості спеціальностей

Індекс спорідненості пари спеціальностей (A, B) розраховується таким чином [11]:

$$J(A,B)=\frac{N_{A\cap B}}{N_A+N_B-N_{A\cap B}}, \quad (1)$$

де N_A – кількість публікацій за спеціальністю A;

N_B - кількість публікацій за спеціальністю B;

$N_{A\cap B}$ - кількість публікацій, які одночасно віднесено як до спеціальності A, так і до спеціальності B.

Проілюструємо застосування формули (1) на прикладі розрахунку спорідненості спеціальностей 4602 та 4608. За спеціальністю 4602 є 316910 публікацій, за спеціальністю 4608 – 239664 публікації. За обома спеціальностями одночасно категоризовано 17757 публікацій. Відповідно рівень спорідненості спеціальностей 4602 та 4608 становить: $J(4602,4608)=$

$$=\frac{17757}{316910+239664-17757}=0.03.$$

Dimensions можна за API [12]. Дані можна отримати використовуючи звичайний HTTP запит, де тілом є текст у форматі Dimensions Search Language [13]. Наприклад, для того, щоб отримати розподіл кількості публікацій за кожною спеціальністю за останні 5 років необхідно відправити запит із таким тілом:

```
search publications
where year in
[2019, 2020, 2021, 2022, 2023]
return category_for_2020
```

Результат повертається у JSON-форматі, в якому для кожної спеціальності вказано кількість публікацій, категоризованих до відповідної спеціальності. Для того, щоб знайти кількість публікацій віднесених одночасно до двох спеціальностей у запиті необхідно додатково вказати ідентифікатор спеціальності. Як відповідь повертається кількість публікацій, які категоризовані одночасно до вказаної спеціальності і

до кожної іншої спеціальності. Приклад такого запиту для спеціальності 3202, яка має внутрішній ідентифікатор 80045, наведено нижче:

```
search publications
where year in
[2019, 2020, 2021, 2022, 2023]
and category_for.id=80045
return category_for_2020
```

Надіславши API-запит для кожної спеціальності отримуємо необхідні початкові дані для розрахунку рівня спорідненості всіх пар спеціальностей. Ранговий розподіл пар спеціальностей за індексом Жаккара (1) наведено на рис. 2. Високу спорідненість має 20 пар спеціальностей, індекс Жаккара для яких перевищує 0.2 (табл. 1). Поміж 20 сильно споріднених пар 3 пари утворено спеціальностями з різних галузей знань. Середня спорідненість має місце для 41 пари спеціальностей (табл. 2). Поміж 41 пари із середньою спорідненістю 9 пар утворено спеціальностями з різних галузей знань. Спорідненість пар із високими та середніми індексами Жаккара варто врахувати під час автоматичного підбору рецензентів чи виявлення схожих наукових та освітніх установ. Можливо, деякий ефект буде і від врахування пар спеціальностей із низьким рівнем спорідненості. Таких пар виявилося 70. Решта пар спеціальностей, а саме $14535-131=14404$ має шумову спорідненість – їхній індекс Жаккара менше 0.05. Поріг щодо шуму встановлено на підставі обчислювальних експериментів [14].

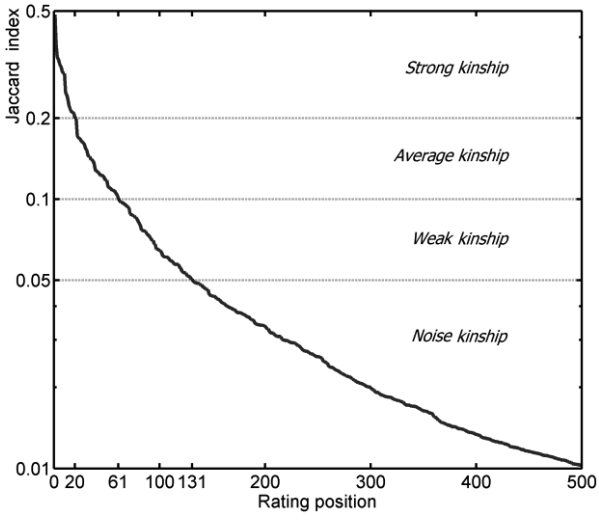

Рис. 2. Фрагмент рангового розподілу спорідненості пар спеціальностей (напівлогарифмічний формат)

Таблиця 1.

Спеціальності з сильною спорідненістю

Спеціальності		Індекс Жаккара
3504	3508	0.485
4703	4704	0.378
3506	3508	0.339
3504	3506	0.332
4008	4009	0.322
4002	4017	0.313
3002	3004	0.309
3901	3903	0.297
5106	5107	0.294
4402	4805	0.293
3503	3506	0.247
4901	4904	0.245
4007	4010	0.236
5106	5110	0.224
3705	3706	0.218
3703	3706	0.212
3703	3705	0.211
4902	5107	0.21
3605	4701	0.208
4004	4011	0.2

Таблиця 2.

Спеціальності з середньою спорідненістю

Спеціальності		Індекс Жаккара
3802	3803	0.199
3302	4005	0.172
3502	3801	0.169
4901	4903	0.168
3501	3502	0.167
3801	3802	0.164
5102	5108	0.163
5004	5005	0.162
4902	4904	0.159
3402	3405	0.154
4006	4613	0.152
4203	4205	0.146
4604	4606	0.144
3103	3104	0.144
5201	5205	0.14
4803	4807	0.14
3304	3509	0.139
3102	3105	0.138
3209	5202	0.129
3705	3709	0.127
4407	4408	0.127
5201	5203	0.126
4605	4606	0.124
3801	3803	0.123
4006	4009	0.123

Спеціальності		Індекс Жаккара
4602	4605	0.122
5101	5109	0.122
3902	3903	0.119
3204	3211	0.118
4804	4807	0.118
3404	3405	0.112
3403	4016	0.111
3406	4016	0.11

Спеціальності		Індекс Жаккара
5202	5204	0.109
4903	4904	0.108
4803	4804	0.108
3702	3709	0.107
3503	4609	0.106
4301	4303	0.105
5002	5003	0.104
3107	3207	0.101

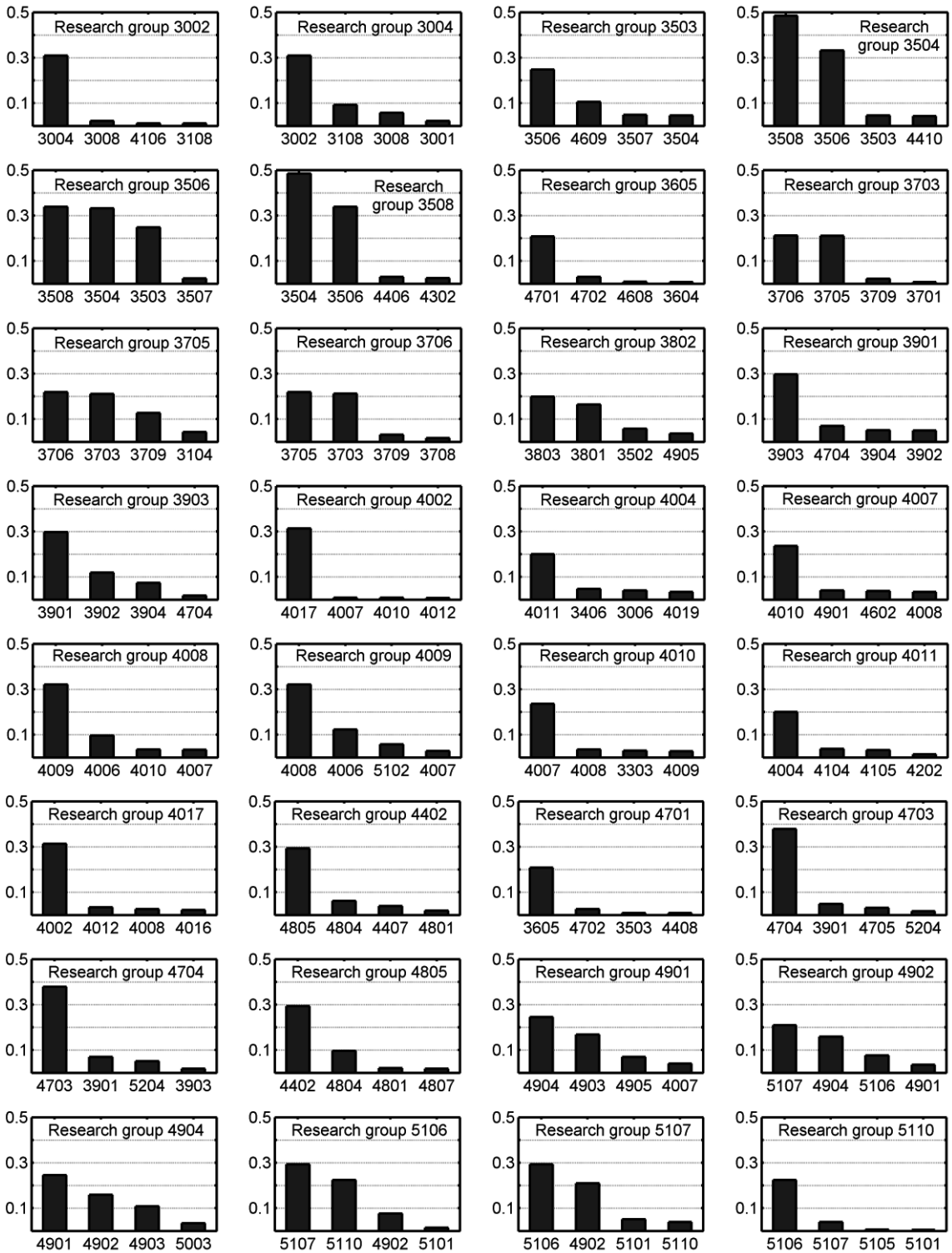

Рис. 3. Діаграми спорідненості спеціальностей з топ-20 за індексом Жаккара

На рис. 3 наведені діаграми спорідненості для 32 спеціальностей, які утворили пари з максимальними індексами Жаккара. Більшість із них мають сильну спорідненість лише з однією спеціальністю. Водночас, 7 спеціальностей сильно взаємодіють з двома спеціальностями. Спеціальність 3506 має високу спорідненість аж із трьома спеціальностями – 3504, 3508 та 3503. Усі вони належать до спільної галузі 35 – Commerce, Management, Tourism and Services.

Висновки

Ідентифіковано рівні спорідненості спеціальностей за системою класифікації наукових спеціальностей ANZSRC-2020. Ідентифікація здійснена з використанням інформаційної системи Dimensions шляхом аналізу 33.8 млн публікацій за 2019–2023 рр. Рівень спорідненості оцінено за індексом Жаккара. Встановлено, що із 14535 можливих пар спеціальностей, лише 131 пара має значнішу спорідненість з індексом Жаккара, що перевищує 0.05. З них для 20 пар спеціальностей рівень спорідненості є високим, а для 61 пари – середнім. Поміж пар із високою спорідненістю переважають спеціальності з однакових галузей, але 3 пари із 20 утворено із спеціальностей з різних галузей.

Отримані оцінки рівня спорідненості спеціальностей можуть бути використані для покращення розв'язання задач автоматичного призначення рецензентів дисертацій, запитів на гранти, рукописів статей, для створення міждисциплінарних PhD-програм, удосконалення системи класифікації наук, відслідковування трендів міждисциплінарних досліджень тощо.

Подяки

Автори висловлюють подяку Digital Science & Research Solutions Inc. за надання доступу до ресурсів Dimensions за проєктом DIM-371.

Література

1. Porter, S. J., Hawizy, L., & Hook, D. W. (2023). Recategorising research: Mapping from FoR 2008 to FoR 2020 in Dimensions. *Quantitative Science Studies*, *4*(1), 127–143. https://doi.org/10.1162/qss_a_00244.
2. *Frascati Manual 2015*. *Frascati Manual 2015*. OECD. https://doi.org/10.1787/9789264268111-ko.
3. Legendre, A. (2019). The development of the Canadian research and development classification. *Knowledge Organization*. International Society for Knowledge Organization. https://doi.org/10.5771/0943-7444-2019-5-371.
4. Wagner, C. S., Roessner, J. D., Bobb, K., Klein, J. T., Boyack, K. W., Keyton, J., Rafols I., Börner, K. (2011). Approaches to understanding and measuring interdisciplinary scientific research (IDR): A review of the literature. Journal of Informetrics, 5(1), 14–26. https://doi.org/10.1016/j.joi.2010.06.004.
5. Dias, L., Gerlach, M., Scharloth, J., & Altmann, E. G. (2018). Using text analysis to quantify the similarity and evolution of scientific disciplines. Royal Society Open Science, 5(1). https://doi.org/10.1098/rsos.171545.
6. Porter, A. L., Cohen, A. S., David Roessner, J., & Perreault, M. (2007). Measuring researcher interdisciplinarity. Scientometrics, 72(1), 117–147. https://doi.org/10.1007/s11192-007-1700-5.
7. Van Noorden, R. (2015, September 16). Interdisciplinary research by the numbers. Nature. Nature Publishing Group. https://doi.org/10.1038/525306a.
8. Braam, R. R., Moed, H. F., & van Raan, A. F. J. (1991). Mapping of science by combined co-citation and word analysis. II: Dynamical aspects. Journal of the American Society for Information Science, 42(4), 252–266. https://doi.org/10.1002/(SICI)1097-4571(199105)42:4<252::AID-ASI2>3.0.CO;2-G.
9. Silva, F. N., Amancio, D. R., Bardosova, M., Costa, L. da F., & Oliveira, O. N. (2016). Using network science and text analytics to produce surveys in a scientific topic. Journal of Informetrics, 10(2), 487–502. https://doi.org/10.1016/j.joi.2016.03.008.
10. Karlovčec, M., & Mladenić, D. (2015). Interdisciplinarity of scientific fields and its evolution based on graph of project collaboration and co-authoring. Scientometrics, 102(1), 433–454. https://doi.org/10.1007/s11192-014-1355-y.
11. Shtovba, S., & Petrychko, M. (2019). Jaccard index-based assessing the similarity of

research fields in dimensions. In CEUR Workshop Proceedings (Vol. 2533, pp. 117–128). CEUR-WS.
12. Dimensions API request. Електронний ресурс. Режим доступу: https://www.dimensions.ai/products/all-products/dimensions-api/ (20.02.2024).
13. Dimensions DSL. Електронний ресурс. Режим доступу: https://docs.dimensions.ai/dsl/ (20.02.2024).
14. Shtovba S., Petrychko M., Shtovba O. Similarity metric of categorical distributions for topic modeling problems with akin categories // CEUR Workshop Proceedings, Vol. 3392 "Proc. of the Sixth International Workshop on Computer Modeling and Intelligent Systems". – 2023. – P. 76-85. DOI: https://doi.org/10.32782/cmis/3392-7.

Одержано: 22.02.2024

Про авторів:

Штовба Сергій Дмитрович,
професор, д. т. н., професор кафедри
інформаційних технологій
Донецького національного університету
імені Василя Стуса.
Кількість наукових публікацій
в українських виданнях – понад 100.
Кількість наукових публікацій
в іноземних виданнях – понад 50.
Індекс Гірша – 8.
https://orcid.org/0000-0003-1302-4899

Петричко Микола Володимирович,
аспірант кафедри комп'ютерних
систем управління Вінницького національного
технічного університету.
Кількість наукових публікацій
в українських виданнях – 7.
Кількість наукових публікацій
в іноземних виданнях – 6.
Індекс Гірша – 3.
https://orcid.org/0000-0001-6836-7843

Місце роботи авторів:

Донецький національний університет
імені Василя Стуса,
600-річчя, 21, 21021, м. Вінниця
email: s.shtovba@donnu.edu.ua
Вінницький національний
технічний університет,
Хмельницьке шосе, 95,
21021, м.Вінниця
email: mpetrychko@vntu.edu.ua

УДК 004.4`2 http://doi.org/10.15407/pp2024.01.086

B.I.Dobrianskyi, I.V. Stetsenko

ARCHITECTURAL FRAMEWORK FOR A UNIFIED BLOCKCHAIN INTERACTION LIBRARY

This research addresses the challenges posed by the fragmented nature of blockchain development tools, presenting a comprehensive exploration of the imperative need for a unified architecture. In response to the growing diversity of blockchain networks, a solution in the form of a unified library based on a versatile and interoperable interface that streamlines interactions across various blockchains is proposed. The current state of blockchain development tools, an overview of existing solutions, formulation of design principles, and functions of proposed unified library are provided. By mitigating the complexities associated with disparate tools, the research aims to enhance the accessibility and efficiency of blockchain development, encouraging collaboration and innovation within the blockchain community.

Keywords: Blockchain development, Interoperability, Decentralized applications, Development tools, Blockchain networks, Software development kits (SDKs), Software Architecture

Introduction

The recent explosion of blockchain technology has created a captivating landscape teeming with innovation and brimming with potential. As mentioned in the survey [1], the use of blockchain has already been extended far beyond “electronic cash”, as initially described by Nakamoto S. [2]. The rapid proliferation has birthed a multitude of diverse blockchains, each boasting unique features, consensus mechanisms, and smart contract functionalities. While the abundance presents developers with exciting opportunities to explore and create, it also presents a significant challenge.

The tools currently available to developers for building upon and interacting with these diverse platforms remain largely limited to specific blockchains. This creates a fragmented development environment, forcing developers to juggle a plethora of libraries, frameworks, and SDKs just to navigate the landscape. This fragmented ecosystem poses a significant barrier to entry for newcomers and seasoned developers alike, hindering efficiency and collaboration within the blockchain development community.

To address the pressing need for standardization and interoperability, this research proposes the development of a unified library. The comprehensive library aims to act as a cohesive interface, empowering developers to seamlessly interact with various blockchains. By providing a standardized, adaptable, and interoperable solution, this research seeks to empower developers by streamlining their workflow, fostering collaboration within the community, and ultimately propelling innovation within the realm of blockchain development.

The following sections describe the challenges presented by the fragmented state of blockchain development tools. After that, existing solutions are examined before articulating the foundational principles and features of the proposed unified library. The aim of the above-mentioned exploration is to provide a comprehensive understanding of the motivation behind and the potential impact of our research on the broader blockchain development community. Looking beyond the proposed solution, the research acknowledges the valuable contributions of existing solutions that play a vital role in the ongoing process of improving interoperability and enhancing developer experience within the blockchain landscape.

1. Essential Components of Unified Blockchain Interface

The unified blockchain library development requires a thorough understanding of its core components, independent of the underlying blockchain technology. From managing accounts to facilitating transactions, these

components are the building blocks of the innovative framework.

To ensure a well-designed architecture, the requirements that govern its construction should be carefully analyzed. The analysis delves into the complexities involved creating a unified interface that seamlessly interacts with diverse blockchains, beginning with the fundamental domain of account management, the cornerstone of any blockchain system.

1.1. Account management

Regarding blockchain networks, one common requirement is the solid management of accounts, which emerges from the role of accounts in the process of assuring secure and authenticated connections. Blockchain security heavily relies on private keys. Each user has a unique digital identity formed by a key pair: a public key (like a public address) and a private key (kept secret). Creating an account with a private key is the entry point for user activity, smart contracts, and transactions on the network. These cryptographic keys, which play the role of digital signatures [3], form the basis of the security architecture of the blockchain networks. The algorithms used may vary, for example, widely adopted cryptocurrencies like Bitcoin leverage the Elliptic Curve Digital Signature Algorithm (ECDSA) [4] for key generation and verification, ensuring a robust foundation for secure transactions. In contrast, Ethereum predominantly utilizes ECDSA but also supports the RSA algorithm in certain contexts [5], providing flexibility in cryptographic operations across its network. Additionally, emerging technologies explore more exotic algorithms, such as the Lamport Signature Scheme [6], showcasing the continuous evolution of cryptographic approaches within diverse blockchain ecosystems.

Creating a security key interweaves the formation of a key pair comprised of a public key visible to everyone and a private key known only to the account owner. Thus, creating an account serves as a starting point for involvement in blockchain activity including user activity and smart contract interactions.

Amongst the diverse functionalities encompassed by blockchain account management, two operations emerge as the most generic and essential: account creation and account retrieval. These operations prove critical in laying the groundwork for secure and decentralized participation within blockchain ecosystem, so they are included in the account management.

Having established the significance of account management, this analysis shifts its focus toward the core entity that underpins blockchain interaction: the account. While the specifics of account implementation vary across different blockchains, the library should bridge this gap by presenting a unified perspective. As per the accounts themselves, their significance extends beyond mere placeholders for cryptographic keys; they serve as dynamic entities within the blockchain ecosystem. Within the unified library, the account encapsulates a suite of functionalities tailored to these entities, fostering an environment that not only ensures secure interactions but also facilitates comprehensive account management.

The most basic functional requirement of the account is the ability to retrieve its balance. This fundamental feature provides users with a real-time snapshot of the account's holdings, detailing the quantity of digital assets or tokens it possesses.

Transaction signing equips developers with the capability to sign transactions securely using the account's private key. This cryptographic signature not only ensures the transaction authenticity, but also safeguards against unauthorized alterations, thereby fortifying the integrity of the blockchain network.

Beyond immediate financial insights, the account encompasses the functionality to query transaction history, which provides a historical record of transactions associated with the account. The comprehensive log offers visibility into past interactions, empowering developers to track and analyze the account activity over time.

To augment the versatility of account interactions, one of the account features is the ability to query metadata associated with it. It includes information such as creation timestamps, account types, custom tags, or

any other pertinent details configured. Metadata querying provides a flexible interface for developers to retrieve contextual information, enhancing the richness of the account management experience.

Moreover, recognizing the dynamic nature of blockchain ecosystems and the diversity among different networks, the account is designed to be extensible. It ensures that the account structure can accommodate additional functionalities specific to particular blockchains. Blockchain networks often introduce unique features or token standards that may require additional methods or attributes within the account representation.

Consider the diversity across blockchain networks such as Ethereum, Solana, and Polkadot, each introducing unique functionalities and adhering to specific standards. The account structure within the unified library is designed to be extensible, allowing developers to seamlessly integrate blockchain-specific features. For instance, in Ethereum, renowned for its extensive smart contract capabilities, the Account might extend functionalities for interacting with sophisticated contract deployment and execution processes. Solana, with its high-performance blockchain, might introduce account-related features pertinent to its unique consensus mechanism and token standards. Similarly, Polkadot's multi-chain architecture might necessitate account-related features aligned with its parachain model.

In essence, the Account class emerges as a multifaceted entity within our unified library, aligning with the diverse needs of developers engaged in blockchain interactions. By encompassing balance retrieval, transaction signing, transaction history querying, and metadata retrieval, the Account class serves as a comprehensive toolkit, fostering an environment where developers can seamlessly manage and leverage the full potential of blockchain accounts within their decentralized applications.

1.2. Block management

At the core of every blockchain lies the foundational building block, quite literally known as a "block." A block represents a unit of data that contains a bundle of transactions, a timestamp, and a reference to the preceding block. This sequential chain of blocks forms the bedrock of blockchain technology, enabling the secure and transparent recording of transactions across a decentralized network.

The significance of a block extends beyond its definition; it serves as a crucial mechanism for achieving consensus and immutability within the blockchain. Each block is intricately linked to its predecessor through cryptographic hashes, creating an indelible chain that resists tampering and ensures the integrity of the entire transaction history.

What sets the concept of a block apart is its universality across diverse blockchain networks. While blockchains may differ in their consensus mechanisms, transaction processing speeds, and smart contract capabilities, the fundamental structure of a block remains remarkably consistent [7]. This uniformity provides an opportune avenue for unification, as the shared characteristics of blocks can be harnessed to create a standardized approach to managing and retrieving blockchain data.

This section delves into the universal nature of blocks, investigating their constituent components and exploring the potential for a unified strategy in managing these foundational elements across diverse blockchain networks. By analyzing the shared characteristics and functionalities of blocks, the groundwork can be laid for the development of a standardized approach to their management, potentially fostering greater interoperability and efficiency within the blockchain ecosystem.

First and foremost, a crucial capability in the toolkit is the ability to fetch the latest block from the blockchain. This feature holds paramount importance because it allows users to stay synchronized with the most recent data on the blockchain by ensuring that the applications are always up-to-date and aligned with the dynamic nature of the blockchain.

Equally vital is the capability to retrieve a specific block using its unique numerical identifier, often referred to as the block number. The significance of this features lies in the ability to grants users precision in data retrieval. Consider a scenario where user wants some specific details from a particular

point in the blockchain's history. By being able to get a block by its number, user can precisely pinpoint and extract the required information. This functionality caters to applications that demand targeted insights into specific blocks, contributing to a more refined and efficient data retrieval process.

Regarding the block itself, it serves as a central entity within our unified framework, facilitating a nuanced exploration of blockchain data.

The following sections investigate the inherent features of blockchain blocks, equipping developers with the capability to effortlessly retrieve crucial metadata associated with each block. This metadata encompasses fundamental details, including the block number, a unique identifier within the blockchain sequence, and the mining time, which provides insights into the block's creation timestamp. These metadata elements collectively offer a comprehensive snapshot, elucidating the block position and temporal relevance within the broader blockchain landscape.

Delving further into the capabilities of the block entity, it extends its functionality to enable the extraction of transactions stored within the block. This feature proves pivotal for developers seeking a detailed understanding of the activities recorded in a specific block. By accessing the list of transactions, developers can explore and leverage the intricacies of each transaction within the block, facilitating a more granular analysis of blockchain data.

In essence, the block entity serves as a versatile tool, offering not only a unique identifier and temporal context through metadata retrieval but also providing access to the transactional intricacies encapsulated within individual blocks. These capabilities enrich the developer's toolkit, fostering a more nuanced and detailed exploration of blockchain data at the block level.

1.3. Transaction management

Navigating the intricacies of blockchain transactions, transaction management takes center stage as a crucial orchestrator. Its role is to simplify the complexities associated with transactions, offering developers a set of powerful functionalities for efficient interaction with the blockchain network.

In terms of transaction retrieval, the transaction manager empowers developers to pinpoint and extract specific transactions. Utilizing identifiers such as transaction hashes, transaction indices, or block references, developers gain flexibility in accessing transactions tailored to their needs. This capability is particularly exemplified in networks like Ethereum, where transaction hashes serve as unique identifiers.

Moving beyond retrieval, the Transaction Manager offers a real-time status check for individual transactions. This feature proves invaluable for developers seeking immediate updates on a transaction's processing status - whether it is pending, successfully executed, or encountering issues. This real-time insight into the transaction lifecycle enhances developers' ability to respond effectively to changing conditions on the blockchain.

Additionally, the transaction manager facilitates historical exploration by enabling developers to query the blockchain for a detailed transaction history associated with a specific account. This functionality is instrumental for applications requiring a comprehensive overview of an account's past transactions. For instance, in decentralized finance applications, users can leverage this capability to review their transaction history seamlessly.

In summary, the Transaction Manager emerges as a versatile guide through the blockchain transactions. Its functionalities, ranging from precise retrieval to real-time status checks and historical exploration, equip developers with the tools needed to navigate the dynamic landscape of decentralized data transactions efficiently.

1.4. Smart contract management

In the realm of blockchain development, smart contracts shine as unique entities injecting programmable capabilities into the decentralized landscape. Unlike blocks, which vary but retain some uniformity, smart contracts stand out distinctly. It's worth noting that not all blockchains, such as Bitcoin and Ripple, natively support smart contracts.

Despite this, smart contracts play a crucial role in blockchain development. They act as architects for self-executing agreements, allowing developers to encode rules directly into the blockchain. This capability, though not universally supported, remains a fundamental tool for developers crafting decentralized applications, which is why smart contract management is included in the library.

The smart contract manager, a pivotal component within the proposed architectural framework, plays a crucial role in facilitating seamless smart contract interactions across diverse blockchain networks. This manager offers two fundamental operations, catering to various blockchain environments.

First of all, it provides a unified interface for retrieving existing smart contracts from the underlying blockchain. This functionality enables developers to interact with deployed smart contracts, including tasks such as reading data, verifying states, and executing predefined operations. By abstracting the complexities of smart contract retrieval, the manager enhances developer efficiency and ensures a consistent experience across different blockchain networks.

In addition to retrieving existing smart contracts, the manager empowers developers to deploy new contracts onto the blockchain. This essential feature facilitates the initiation of decentralized applications and the implementation of custom smart contract logic. The manager abstracts the intricacies of the deployment process, offering developers a streamlined mechanism to bring their smart contract creations to life on the blockchain, regardless of the underlying blockchain's specifications.

Building upon the foundation laid by the smart contract manager, the smart contract entity inherits the ability to read data directly from the smart contract it represents. This continuity ensures that developers maintain real-time access to crucial information stored within the decentralized ledger. The entity becomes a conduit for understanding the evolving state and conditions of the contract, fostering an environment of informed decision-making.

As an evolution of transactional capabilities, the smart contract entity empowers developers to submit smart contract transactions to the blockchain. This functionality signifies an active engagement in the execution of predefined functions within the smart contract. Developers navigate the decentralized landscape with a refined ability to participate in the decision-making process encoded within the blockchain.

Extending the narrative of event-driven architecture, the smart contract entity allows developers to seamlessly subscribe to events emitted by the represented smart contract. This continuity in observability enhances transparency, ensuring external entities, for example other smart contracts or off-chain systems, can monitor and react to changes within the contract. The smart contract entity becomes a vital component in fostering a collaborative and responsive blockchain ecosystem.

1.5. Plugin system

A key component of the architecture is a robust plugin system. Recognizing the dynamic and evolving nature of blockchain ecosystems, the plugin system allows for adaptability and extensibility. It enables:

- Streamlined integration of new features: plugins can be added without modifying the core library, keeping development efficient.
- Empowerment for the developer community: developers can contribute and enhance the library's capabilities, fostering collaboration and innovation.

However, achieving true unification requires acknowledging the inherent diversity of blockchain networks. The proposed solution addresses this by emphasizing the use of blockchain-specific plugins. These plugins cater to the unique characteristics and functionalities of each blockchain, ensuring:

- Preservation of individual blockchain design: developers can leverage the full potential of each network without compromising its integrity.
- Optimal performance: plugins allow fine-tuning the library's functionality for specific blockchains, maximizing efficiency.

Balancing standardization and customization are crucial within this framework. Blockchain-specific plugins acknowledge the value of individual network characteristics while maintaining a unified approach through the core library interface. This embrace of diversity reflects the richness and innovation present in the blockchain landscape.

Moving forward, the proposed architecture provides a flexible and adaptable foundation for blockchain development. It empowers developers with a unified interface while acknowledging the unique needs of individual blockchain networks, paving the way for a more collaborative and future-proof development landscape.

2. Results

Continuing upon the established conceptual foundations, this section unveils the architectural design of the proposed unified library. Figure 1 presents a structure diagram of this architecture.

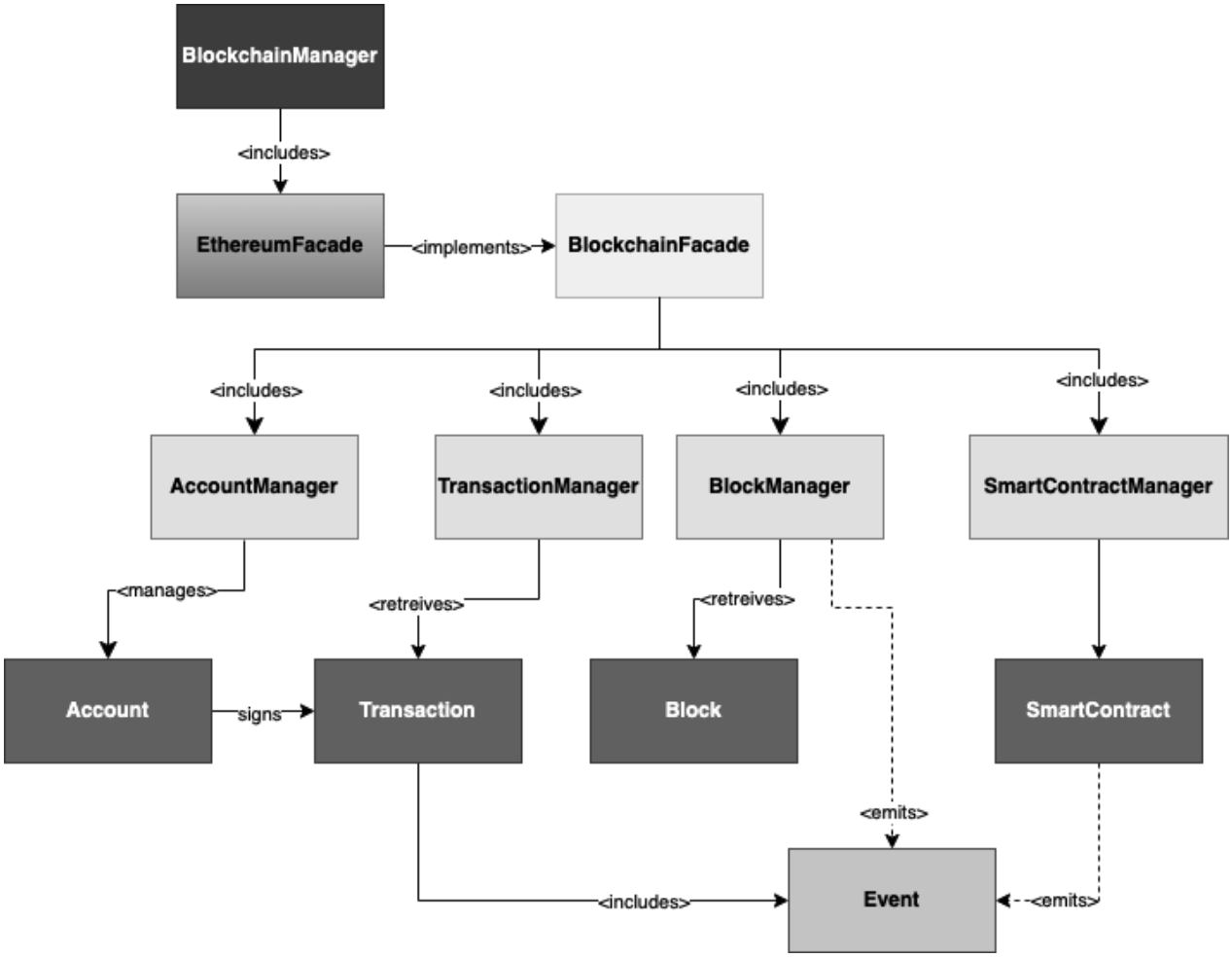

Fig.1. Structure Diagram of the library

The architecture encapsulates key components, allowing for seamless interactions with various blockchain networks. A brief overview of the major components is provided below.

The blockchain manager serves as an entry point to the library. It can include one or more blockchain facades, that are implemented as separate libraries (plugins) for each blockchain.

Transaction Manager stands as a central hub for retrieving, monitoring, and exploring transactions within the blockchain.

Transaction represents individual transactions, this entity encapsulates details, signing mechanisms, verification, and any additional metadata associated with a transaction.

Block Manager serves as a linchpin for managing blocks, the Block Manager of-

fers functionalities such as retrieving specific blocks by number, obtaining the latest block, and navigating through blocks with pagination and filtering. It also emits related events, for example mining of a new block.

The Block entity encapsulates metadata retrieval, facilitating the extraction of transactions, and ensuring extensibility for blockchain-specific features.

Account Manager is responsible for simplifying account-related interactions, the Account manager provides methods for creating and retrieving accounts, signing transactions, and managing account-specific functionalities.

The Account entity represents individual accounts within the blockchain, offering functionalities like balance retrieval, transaction signing, checking transaction history, and querying metadata.

Smart contract manager is responsible for deployment and retrieval of smart contracts. The Smart Contract entity itself is responsible for reading data from the smart contract, calling methods on smart contracts and listening to the events emitted by the smart contract.

This architecture is designed with versatility and extensibility in mind. Its strength lies in providing developers with a unified interface for blockchain interactions, reducing the complexities associated with navigating diverse blockchain networks. The modular structure allows for easy integration and adaptation to specific blockchain characteristics, ensuring a robust foundation for blockchain development. With a foundational understanding of the architecture now established, the following analysis covers the advantages and potential applications that make this design a compelling choice for blockchain development.

2.1. Implementation

The unified blockchain interaction library has been successfully implemented using JavaScript and TypeScript to support the diverse needs of developers engaged in blockchain interactions. The decision to employ these languages is rooted in their widespread adoption within the blockchain development community, serving as the go-to choices for both backend and frontend development. This approach ensures a seamless integration of the library into various projects, maintaining compatibility across different layers of the software stack.

The Ethereum and Solana plugins were also crafted in TypeScript using web3.js and @solana/web3.js libraries correspondingly, ensuring consistency and ease of use across the entire ecosystem. The language choice aligns with the prevalent practices in blockchain development, where JavaScript and TypeScript stand out as the primary languages for facilitating comprehensive blockchain interactions across different platforms.

2.2. Pros and potential use cases

Versatility across blockchains. One of the notable strengths of this architecture lies in its adaptability to a variety of blockchain networks. The modular design allows developers to seamlessly integrate the library into diverse blockchain ecosystems, mitigating the challenges posed by the heterogeneity of blockchain technologies.

Reduced development complexity. By providing a unified interface for blockchain interactions, our architecture significantly reduces the development complexity associated with navigating different blockchain networks. Developers can leverage consistent methods and entities across various blockchains, streamlining the development process and fostering code reusability.

Facilitated cross-blockchain development. The modular structure and unified interface empower developers to engage in cross-blockchain development more efficiently. Whether working with Ethereum, Solana, Polkadot, or other blockchains, the proposed design simplifies the development of applications that seamlessly interact with multiple blockchain networks.

Extensibility for blockchain-specific features. The architecture's extensibility ensures that it can accommodate blockchain-specific features seamlessly. For instance, if a blockchain introduces new metadata or unique transaction types, the library's extensible entities, for example the Transaction and

Account, can readily adapt without requiring a major overhaul of the existing codebase.

2.3. Challenges and considerations

While our architecture presents numerous advantages, it is essential to acknowledge potential challenges and considerations.

Blockchain-specific limitations. Certain blockchain networks may have unique limitations or characteristics that might not align perfectly with the unified interface. Developers should be aware of such nuances and handle them appropriately when working with specific blockchains.

Performance considerations. The architecture's adaptability across various blockchains may introduce slight performance trade-offs. Developers should carefully assess performance requirements and optimize implementations accordingly. However, this also heavily depends on the implementation itself.

3. Related works

From specialized libraries tailored for specific blockchain networks to comprehensive frameworks designed for cross-chain compatibility, the variety in the approaches mirrors the continuous evolution of blockchain technology. The research through these related works unveils the multifaceted efforts that shape the current and future landscapes of blockchain development.

In the following exploration of related works, 3 distinctive solutions that have made significant strides in simplifying and unifying blockchain interactions are spotlighted. These solutions, each with their unique approach, contribute to the broader narrative of enhancing developer experiences within the blockchain landscape.

3.1. Web3.js

Web3.js stands as a JavaScript library specifically tailored for EVM blockchains, providing developers with a streamlined interface for interacting with the Ethereum blockchain. It abstracts away the complexities of Ethereum's native RPC (Remote Procedure Call) and JSON-RPC protocols, offering a simplified programming model.

Designed with a focus on Ethereum, Web3.js provides comprehensive functionalities for smart contract interactions, transaction management, and blockchain data retrieval specific to the Ethereum network.

Being a widely adopted library, Web3.js benefits from an active and vibrant community, ensuring continuous updates, bug fixes, and the availability of extensive documentation.

It is important to note, however, that Web3.js is primarily applicable for EVM-compatible networks, and its features are optimized for the Ethereum ecosystem.

3.2. Cosmos SDK

The Cosmos SDK takes a broader approach by offering a framework for building multi-blockchain applications. It allows developers to create their customized blockchains (Zones) with specific functionalities, while the Cosmos Hub acts as a secure and interoperable hub connecting different Zones [9].

Cosmos SDK focuses on enabling interoperability between blockchains. The Hub-and-Zone architecture allows for secure communication between different blockchains, fostering a vision of an interconnected blockchain ecosystem.

Developers using Cosmos SDK have the flexibility to design and launch their blockchains with tailored features, ensuring versatility in addressing specific use cases.

3.3. Polkadot and Parachains

Polkadot introduces a novel concept of parachains, offering a framework for building customized blockchains that connect to the Polkadot relay chain. Parachains run in parallel, fostering scalability and interoperability across the Polkadot network [10].

The use of parachains in Polkadot enables horizontal scalability, allowing multiple blockchains to run concurrently. This approach addresses scalability concerns and promotes efficient resource utilization.

By connecting to the Polkadot relay chain, parachains benefit from enhanced security and interoperability, creating a cohesive ecosystem where different blockchains can seamlessly interact.

3.4. Challenges

This survey reveals a common thread – interoperability is often confined to specific blockchain ecosystems, lacking a universal solution that spans the entirety of the blockchain landscape. While these existing libraries and frameworks excel within their designated environments, the quest for a unified architecture for blockchain interaction persists.

A constantly developing landscape of blockchain innovation necessitates a comprehensive solution. Analyzing the process of designing a unified library for blockchain interaction revealed both challenges and opportunities. By envisioning an architecture that transcends individual blockchain ecosystems, the goal is to foster an interoperable development landscape, empowering developers across diverse networks. With wider adoption and expansion of blockchain usage, the need for a universal framework becomes increasingly evident, paving the way for a more interconnected and collaborative future in decentralized development.

Conclusions

This research addresses the challenge of fragmented blockchain development tools by proposing a novel unified library architecture. This architecture aims to bridge the gap by providing developers with a standardized and interoperable interface, facilitating seamless interactions across diverse blockchain networks.

The proposed solution presents a step forward in blockchain development, offering the following advantages:

− Enhanced developer efficiency. By providing a unified interface, the architecture significantly reduces the complexity caused by navigating and interacting with various blockchain ecosystems.

− Promoted cross-chain collaboration. Whether building applications that interact with Ethereum, Solana, Polkadot, or other emerging networks, the architecture simplifies communication and data exchange, fostering collaboration and innovation across the broader blockchain landscape.

− Future-proof design. Blockchain-specific features can be seamlessly integrated through adaptable entities within the library, ensuring it remains relevant and adaptable as new protocols and functionalities emerge.

Potential challenges requiring further investigation include:

− Addressing blockchain-specific nuances. While the unified interface offers significant advantages, particular blockchain networks may possess unique limitations or characteristics that it does not cover. Developers should know these nuances and implement appropriate workarounds interacting with specific blockchains.

− Balancing performance and versatility. The architecture adaptability across various blockchains might necessitate slight performance trade-offs.

Based on the proposed architectural framework, the future improvements are the following:

– conducting rigorous testing and performance evaluations across diverse blockchain networks to identify and address potential limitations;

– expanding the library's capabilities by developing and integrating plugins for a wider range of blockchain networks;

– investigating and integrating emerging interoperability protocols and solutions to further enhance seamless communication across blockchain ecosystems;

– conducting in-depth research on optimizing the library's internal data structures and algorithms to ensure efficient and scalable interactions across diverse blockchain networks.

By taking these steps, the proposed unified library architecture can evolve into a robust and versatile solution, empowering developers to navigate the ever-evolving landscape of blockchain development with greater efficiency and fostering a more collaborative and innovative future for decentralized technologies.

References

1. Al-Jaroodi J., Mohamed N. (2019), 'Blockchain in Industries: A Survey', *IEEE Access*, vol. 7, pp. 36500-36515.
2. Nakamoto S. 'Bitcoin: A Peer-to-Peer Electronic Cash System', available at https://bitcoin.org/bitcoin.pdf

3. Zhang, R., Xue, R., Liu, L. (2019), 'Security and Privacy on Blockchain', *ACM Computing Surveys*, 52, 3, Article 51.
4. Johnson, D., Menezes, A., Vanstone, S. (2001), 'The Elliptic Curve Digital Signature Algorithm (ECDSA) ', *International Journal of Information Security*, 1, 36–63, https://doi.org/10.1007/s102070100002
5. Shamir A., Rivest, R., Adleman L. (1978), 'A method for obtaining digital signatures and public-key cryptosystems', *Communications of the ACM*, 21 (2), 120–126.
6. Pandey, S., Behl, R. & Sinha, A. (2023), 'Decentralized blockchain-based security enhancement with lamport merkle digital signature generation and optimized encryption in cloud environment', *Multimedia Tools and Applications,* doi: 10.1007/s11042-023-17365-8.
7. Clincy, V., Shahriar, H. (2019), 'Blockchain Development Platform Comparison', *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, doi: 10.1109/COMPSAC.2019.00142.
8. Web3.js framework. Available at https://github.com/web3/web3.js (Accessed 28 February 2024).
9. Kwon, J., Buchman, E. (2016). Cosmos: a network of distributed ledgers. Available at https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md (Accessed 28 February 2024).
10. Wood, G. (2016). Polkadot: Vision for a heterogenous multi-chain framework. Available at https://assets.polkadot.network/Polkadot-whitepaper.pdf (Accessed 28 February 2024).

Received: 01.03.2024

***Про авторів*:**

Добрянський Богдан Ігорович,
магістрант другого року навчання
Національного Технічного Університету
України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – 1.
Кількість наукових публікацій
в зарубіжних виданнях – 0.
Індекс Гірша – 0.
https://orcid.org/0009-0003-4797-1947

Стеценко Інна Вячеславівна,
доктор технічних наук, професор,
професор кафедри інформатики
та програмної інженерії НТУУ
“КПІ імені Ігоря Сікорського”.
Кількість наукових публікацій
в українських виданнях – понад 140.
Кількість наукових публікацій
в зарубіжних виданнях – 16.
Індекс Гірша – 11.
http://orcid.org/ 0000-0002-4601-0058

Місце роботи авторів:

Національний технічний університет
України «Київський політехнічний
інститут імені Ігоря Сікорського»,
03056, м. Київ,
проспект Берестейський 37.
Тел.: (044) 236-9651
e-mail: bogdan.dobryanskiy@jcloud.com,
stiv.inna@gmail.com

УДК 004.056 http://doi.org/10.15407/pp2024.01.096

Ф.О. Коробейніков

КОНЦЕПТУАЛЬНА РАМКА РЕЗИЛЬЄНТНОЇ ПОВЕДІНКИ ІНФОРМАЦІЙНИХ СИСТЕМ

Пропоноване дослідження присвячене розробленню концептуальної рамки для опису резильєнтної поведінки інформаційних систем, яка визначає основні операційні фази реакції систем у контексті протидії реалізованим ризикам у момент інциденту. Фази включають у себе як заходи попередньої підготовки та безпосередньої протидії загрозам, так і механізми відновлення та адаптації після інцидентів. Досягнення зазначеної мети передбачає виявлення основних цілей і задач резильєнтності, які безпосередньо пов'язані з процесами нейтралізації потенційних загроз і впливають на стратегію управління ризиками, і зумовлюють необхідність всебічного аналізу критично значущих функцій організацій. У процесі дослідження особливу увагу було приділено створенню моделей резильєнтних і нерезильєнтних сценаріїв реакцій на інцидент та виокремленню поведінкових відмінностей. Застосування дельта-функції дало змогу деталізувати фазу резильєнтної відповіді на загрози, включно з етапами адаптації та подальшого відновлення. Дослідження робить значний внесок у сферу інформаційної безпеки, пропонуючи багаторівневий підхід до підвищення резильєнтності інформаційних систем.

Ключові слова: резильєнтність, захист інформації, інформаційна безпека

Вступ

Розбудова фреймворку підвищення резильєнтності будь-якої інформаційної системи, незалежно від її специфіки - чи то система захисту інформації, чи то система обробки та зберігання інформації, що належить організації, місії, або спільноті, - розпочинається з формулювання ключових цілей і задач. Процес їх визначення тісно пов’язаний зі стратегією управління ризиками організацій і вимагає глибокого розуміння того, що саме має бути захищено і які саме загрози можуть впливати на критичні функції організацій і пов’язані з ними інформаційні системи. Метою даного наукового дослідження є розробка концептуальної рамки резильєнтної поведінки інформаційних систем, яка не тільки відповідає вимогам резильєнтної парадигми, а й вміщує методи ідентифікації, ренжування та управління етапами протидії загрозам, віднесеним до категорії критичних, враховуючи їхнє співвідношення з цілями і задачами резильєнтності.

Цілі резильєнтності

Кожна ціль резильєнтності - це твердження високого рівня, що фокусується на одному з аспектів (наприклад, передбачити, витримати, відновити, адаптуватися) у визначенні резильєнтності [1].

На відміну від цілей інформаційної безпеки, які зосереджені на захисті конфіденційності, доступності та цілісності інформаційних активів, цілі резильєнтності виходять із передумови, що реалізація загроз є, хоча й небажаним, але неминучим елементом життєвого циклу системи. Ця концептуальна різниця підкреслює, що в той час як інформаційна безпека прагне запобігти реалізації загроз, резильєнтність визнає і приймає можливість їх виникнення, акцентуючи увагу на збереженні працездатності та швидкому відновленні після порушень.

Цілі резильєнтності, таким чином, визначають принципово інший підхід до управління ризиками, який передбачає розробку стратегій і рішень, які сприяють адаптивності системи в умовах мінливого середовища і несподіваних подій. Це не означає відмову від заходів із забезпечення безпеки, але підкреслює важливість додаткових заходів, спрямованих на забезпечення працездатності критичних функцій системи, коли стандартні методи безпеки виявляються неефективними [2].

Сукупно всі цілі резильєнтності націлені на протидію критичним ризикам і гарантоване підтримання працездатності основних функцій організації, місії або

системи. Кожна з цілей може бути співвіднесена з одним із ключових етапів протидії ризику, загроза якого реалізується або може бути реалізована: підготовка, протидія або абсорбування, відновлення та адаптація.

Використання дельта-функції для опису етапів протидії інциденту

На малюнку (Рис.1) інцидент представлений як миттєве збурення, яке можна змоделювати за допомогою дельта-імпульсу. Ця подія викликає негайне зниження продуктивності системи.

Дельта-імпульс, у рамках цього дослідження пропонується використовувати як для деталізації цілей резильєнтності, так і в контексті розробки моделі їхньої взаємодії. Ця концепція являє собою ідеалізований опис миттєвих, короткострокових подій, що мають високий рівень впливу. Це особливо актуально для сценаріїв критичних HILP-ризиків, які підлягають опрацюванню відповідно до парадигми резильєнтності.

Класична концепція дельта-імпульсу часто асоціюється з дельта-функцією Дірака [3]. У цьому контексті δ-імпульс є ідеалізованим, миттєвим, нескінченно коротким і сильним сигналом або впливом, який відбувається в певний момент часу.

Особливості дельта-імпульсу:

- Миттєвість: дельта-імпульс відбувається за нескінченно короткий проміжок часу.
- Локалізація в часі: Імпульс має місце в конкретний момент часу T, і поза цим моментом його вплив дорівнює нулю.
- Нескінченна амплітуда: У момент імпульсу його амплітуда прагне до безкінечності, але так, що інтеграл від імпульсу за часом залишається скінченним і зазвичай нормалізується до одиниці.

Дельта-функція Дірака $\delta(t-T)$, визначається такими властивостями:

- $\delta(t-T) = 0$ для всех $t \neq T$
- $\int_{-\infty}^{+\infty} \delta(t-T)dt = 1$

Дельта-функція може бути використана в контексті резильєнтності для моделювання миттєвих подій або шоків, що впливають на систему. У цьому контексті вона ілюструє критичні інциденти, які впливають на систему раптово і з сильним ефектом, але за дуже короткий проміжок часу:

- Моделювання зовнішніх шоків: Дельта-функція може бути використана для представлення раптових подій, таких як

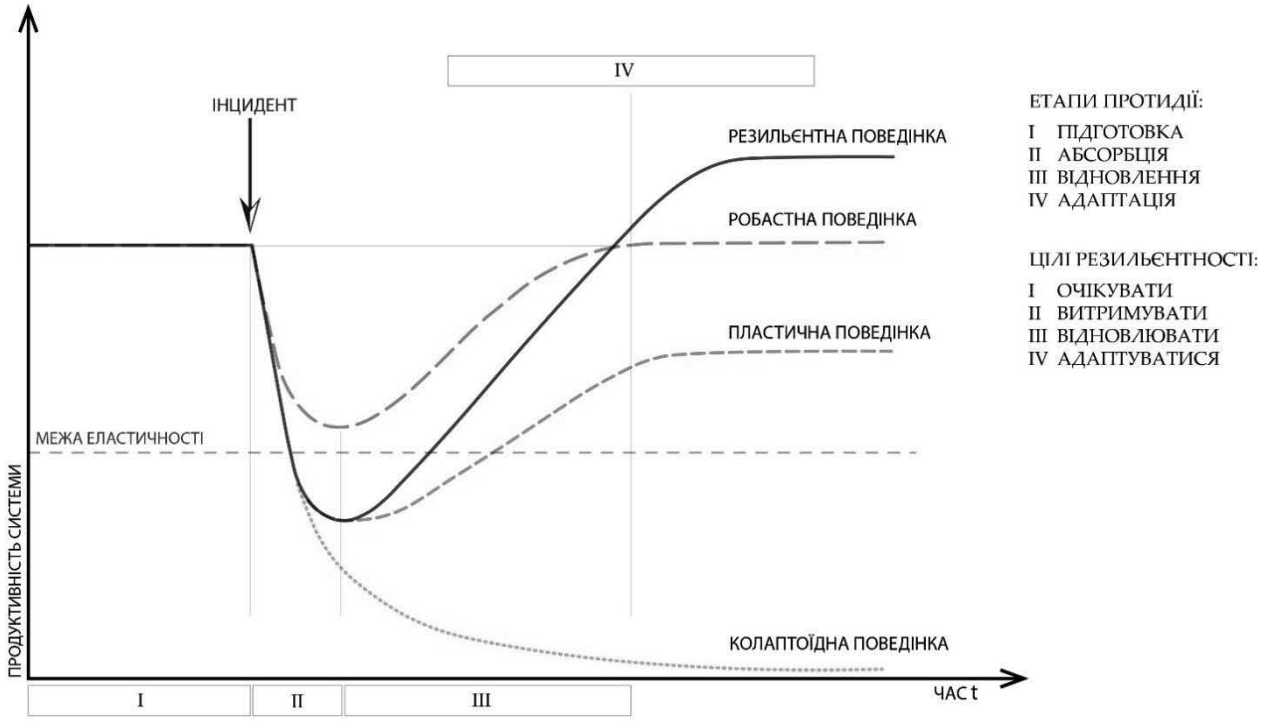

Рис. 1 Співвіднесення цілей резильєнтності і етапів протидії ризику

природні катастрофи, технологічні збої, економічні кризи та інші критичні інциденти. Ці події можуть бути представлені як миттєві імпульси, що впливають на систему.

- Аналіз реакції системи: Використання дельта-функції дає змогу аналізувати, як система реагує на миттєві збурення. Це може допомогти у визначенні ефективності заходів резильєнтності, таких як здатність системи до швидкого відновлення після шоку.
- Інтеграція з іншими моделями: Дельта-функція може бути інтегрована з іншими математичними моделями для створення комплексної моделі резильєнтності, що враховує як миттєві збурення, так і тривалі процеси відновлення та адаптації.

Важливі зауваження:

- Дельта-функція - це ідеалізована математична абстракція. Реальні події можуть мати більш тривалий вплив і не завжди точно відповідати миттєвому імпульсу.
- Реальні системи часто мають складні взаємодії та залежності, які можуть бути не повністю охоплені спрощеним поданням через дельта-функцію.

Таким чином, хоча дельта-функція є корисним інструментом для моделювання деяких аспектів резильєнтності, її слід використовувати з розумінням притаманних їй обмежень і часто в поєднанні з іншими методами та моделями.

Розглянемо дві моделі реакції системи на реалізацію критичного ризику. Одна з моделей є нерезильєнтною, а інша - резильєнтною, що включає всі парадигматичні етапи реагування на інцидент, ототожнювані з цілями резильєнтності. Обидві ці моделі використовують дельта-функцію для представлення інциденту і застосовують додаткові математичні функції для опису основних етапів реакції системи на цю подію.

Припустимо, що аналізується реакція супутникової системи передачі даних на миттєвий зовнішній вплив, наприклад, на потужний сонячний спалах. Це збурення може являти собою, наприклад, великий технологічний збій критичних функцій системи внаслідок коронального викиду маси на Сонці з індексом Dst -1500 нТл.

Сценарії нерезильєнтної поведінки

У контексті моделювання нерезильєнтної системи, яка стикається з миттєвим, коротким, але критичним інцидентом, модель може бути сформульована таким чином:

$$S(t) = P + \delta(t - T_0) \cdot I_0 + R(t)$$

де:

$S(t)$ - стан системи в часі t.

P - рівень підготовки системи до інциденту, що являє собою постійне значення, яке відображає базовий стан системи.

$\delta(t - T_0)$ - дельта-функція Дірака, використовувана для моделювання миттєвого впливу інциденту в момент часу T_0.

I_0 - інтенсивність інциденту, що відображає величину його миттєвого впливу на систему.

$R(t)$ - функція відновлення, що описує процес відновлення системи після інциденту. Ця функція може бути лінійною або експоненційною, залежно від характеру системи та її здатності до відновлення.

Наприклад:

$$R(t) = -R_0 \times e^{-k(t-T_r)} \text{ для } t \geq T_r$$

У цій моделі миттєвий критичний інцидент чинить істотний і негайний вплив на систему в момент часу, що може викликати короткочасне, але серйозне порушення.

Відновлення $R(t)$ являє собою процес, протягом якого система намагається повернутися до свого початкового стану.

Однак у контексті нерезильєнтної системи, цей процес може бути недостатнім або затяжним і відображає обмежену здатність системи до ефективного відновлення і призвести до реалізації трьох сценаріїв поведінкових реакцій:

- *Робастна поведінка* [4]: Система, що демонструє робастну поведінку, зберігає свою функціональність на первісному рівні навіть після впливу зовнішнього шоку, не виявляючи помітного зниження продуктивності - не опускаючись нижче рівня еластичності. Це свідчить про достатність заходів підготовки до інциденту - про наявність у системі вбудованої стійкості або надмірності, що дають змогу абсорбувати раптові

впливи без значних порушень у її роботі. Така система зазвичай характеризується високим ступенем надійності, завдяки чому вона здатна підтримувати критично важливі функції навіть в умовах екстремальних збурень.

- *Пластична поведінка* [5]: У разі пластичної поведінки, система відчуває тимчасове зниження продуктивності у відповідь на шок, після чого відбувається її часткове відновлення. Однак система не здатна повністю відновитися до рівня продуктивності вихідного стану, що вказує на відсутність достатніх механізмів підготовки, адаптації та відновлення.

- *Колапсоідна поведінка* (Collapsing Behavior): Означений тип поведінки характеризується нездатністю системи впоратися з раптовим шоком, унаслідок чого вона втрачає свою функціональність. Така поведінка може призвести до повного краху системи або вимагати фундаментальної реорганізації та перебудови для відновлення її працездатності. Це підкреслює критичну важливість включення заходів запобігання катастрофічним подіям у загальну стратегію управління ризиками та резильєнтності.

Тобто, аналіз нерезильєнтних реакцій систем на шокові події дає змогу глибше зрозуміти рівень їхньої гарантоспроможності і виявити потенційно слабкі місця в структурі та функціонуванні, що є ключовим аспектом ефективного управління ризиками та стратегії резильєнтності.

Резильєнтний сценарій реакції

В рамках концептуалізації моделі системи, що зазнає впливу миттєвого, короткочасного, проте критично значущого інциденту, представлення моделі передбачає систематичний і всебічний підхід до моделювання резильєнтності, заснований на точному кількісному та якісному аналізі динаміки системи в умовах невизначеності й потенційних ризиків, і може бути здійснене в такий спосіб:

$$S(t) = P + \delta(t - T_0) \cdot I_0 + W(t) \cdot H(t - T_w) + R(t) \cdot H(t - T_r) + A(t) \cdot H(t - T_a)$$

де:

$S(t)$ - стан системи в часі t.

P - рівень підготовки системи до інциденту, що являє собою постійне значення, яке відображає базовий стан системи.

$\delta(t - T_0) \cdot I_0$ - модель миттєвого інциденту в момент часу T_0 , де I_0 - інтенсивність впливу інциденту.

$W(t)$ - функція протидії або абсорбції інциденту.

$H(t - T_x)$ - ступінчаста функція Хевісайда, що активує відповідні процеси в момент часу T_x

$R(t)$ - функція відновлення, що починається після часу T_r і триває до досягнення стабільного стану.

$A(t)$ - функція адаптації, що починається після часу T_a і відображає поліпшення системи на основі досвіду інциденту.

В цій моделі ступінчаста функція Хевісайда, також відома як одинична ступінчаста функція, - це математична функція, що має значення нуль до певного моменту (назвемо його t_0) і значення один після цього моменту [6]. У математичному записі:

$$\begin{cases} 0, & \text{якщо } t < t_0 \\ 1, & \text{якщо } t \geq t_0 \end{cases}$$

Ця функція використовується для моделювання ситуацій, коли відбувається раптова зміна стану в певний момент часу. У контексті моделі резильєнтності, ступінчасті функції Хевісайда використовували для активації певних процесів тільки після настання певних подій.

T_w , T_r , T_a являють собою часові точки початку процесів абсорбції, відновлення та адаптації відповідно.

Протидія/абсорбція $W(t)$ відображає заходи, яких вживає система негайно після інциденту для пом'якшення його впливу.

Відновлення $R(t)$ показує, як система повертається до нормальної функціональності та можливо досягає поліпшеного стану після інциденту.

Адаптація $A(t)$ підкреслює процес поліпшення системи у відповідь на досвід, отриманий під час інциденту, посилюючи загальну резильентність системи.

Модель охоплює основні етапи протидії та ілюструє, як резильєнтна система може ефективно реагувати на миттєві

інциденти, швидко відновлюватися й адаптуватися, щоб поліпшити свою здатність протистояти майбутнім загрозам.

Однак не всі інциденти в контексті резильєнтності, особливо ті, що стосуються кіберпростору, можуть бути адекватно представлені через дельта-функцію Дірака, яка символізує миттєвий вплив.

Наприклад, APT-атаки (Advanced Persistent Threats) зазвичай розвиваються поступово і можуть залишатися непоміченими протягом тривалого часу, перш ніж їхні наслідки стануть явними.

У таких випадках інцидент характеризується тривалим періодом, протягом якого загроза поступово посилюється, і може не бути виявлена доти, доки не завдасть значної шкоди.

Замість використання дельта-функції для представлення APT-атак, ми можемо використовувати функцію, яка збільшується з часом або змінюється залежно від виявлення і впливу атаки. Наприклад, можна використовувати безперервну функцію $I(t)$, яка починається з моменту T_0 і збільшується з часом:

$$\begin{cases} a \cdot (t - T_0)^b, & \text{якщо } t \geq T_0 \\ 0, & \text{якщо } t < T_0 \end{cases}$$

де: a і b - параметри, що визначають швидкість зростання та інтенсивність впливу атаки. T_0 - час початку інциденту або атаки. t - поточний час.

Інтерпретація: До T_0 інцидент ще не почався, і його вплив на систему дорівнює нулю. Після T_0 інцидент починається і його вплив на систему поступово збільшується. Степенева функція $(t - T_0)^b$ дає змогу моделювати різні сценарії розвитку інциденту - від тих, що повільно наростають, до більш стрімких.

Припустимо, що на кіберсистему було здійснено APT-атаку в момент часу T_0. Формула для $I(t)$:

$$I(t) = a \cdot (t - T_0)^b, \text{ якщо } t \geq T_0$$

До T_0 атака ще не почалася, і її вплив на систему відсутній. У Момент T_0 і після - атака починається. Спочатку її вплив може бути незначним, але з часом він збільшується. Наприклад, зловмисники можуть спочатку отримати доступ до некритичних частин системи, але поступово проникають глибше, отримуючи доступ до важливіших даних або ресурсів.

Параметри a і b: Параметр a: Визначає початковий рівень впливу атаки. Якщо a малий, початковий вплив атаки невеликий. Параметр b: Визначає, наскільки швидко посилюється вплив атаки. Більше значення b означає більш швидке зростання загрози.

Наприклад, якщо атака почалася в момент $T_0 = 5$ (наприклад, 5 днів після початку спостереження), з параметрами $a = 0.1$ і $b = 2$, то через 10 днів після початку спостереження ($t = 10$) вплив атаки буде:

$$I(10) = 0.1 \cdot (10 - 5)^2 = 0.1 \cdot 25 = 2.5$$

Це означає, що вплив атаки на систему посилюється з часом, що вимагає безперервного моніторингу та переосмислення стратегії застосування адаптивних заходів реагування для забезпечення ефективної відповіді на загрозу.

Інтегрувавши функцію $I(t)$, яка описує лонгітюдний перебіг інциденту в запропоновану раніше модель резильєнтності, що містить етапи підготовки, реагування/абсорбції, відновлення та адаптації, отримаємо:

$$S(t) = P + I(t) \cdot H(t - T_0) + W(t) \cdot H(t - T_w) + R(t) \cdot H(t - T_r) + A(t) \cdot H(t - T_a)$$

Ця інтегрована модель враховує як миттєві, так і тривалі загрози, ілюструючи динаміку реакції резильєнтної системи на різноманітні інциденти та її здатність до відновлення й адаптації.

Серед усіх розглянутих моделей поведінки системи у відповідь на різні сценарії впливу критичних загроз, резильєнтна модель, що характеризується адаптивною поведінкою, демонструє найбільшу ефективність. Ця модель виходить за рамки простого відновлення після шоку, передбачаючи активну адаптацію системи, що передбачає поліпшення її функцій і продуктивності через процеси навчання і самоорганізації. Адаптивна поведінка системи передбачає розробку і впровадження нових структур, функцій, зворотних зв'язків і джерел залучення ресурсів, що приводить не тільки до відновлення, а й до фундаментальної

трансформації системи, зміцнюючи її стійкість до майбутніх шоків і загроз.

Особливої значущості резильєнтна модель набуває в довгостроковій перспективі, оскільки вона сприяє сталому розвитку і підвищенню загальної здатності системи адаптуватися до мінливих умов і непередбачених обставин. Системи, що володіють адаптивною поведінкою, не тільки ефективно долають поточні проблеми, а й безперервно розвиваються, посилюючи свою резильєнтність протягом усього життєвого циклу.

Детальний аналіз резильєнтної поведінки системи виявляє наявність чотирьох критичних і невід'ємних етапів, які корелюють з основними цілями резильєнтності, визначеними відповідно до її концептуальної парадигми. Ці етапи включають: підготовка (ціль - очікувати), абсорбція (витримати), відновлення (відновити) та адаптація (адаптуватися) [7]. Кожна з цих цілей характеризується унікальними атрибутами та функціями:

- *Очікувати*. Ітеративне вдосконалення методик виявлення та опрацювання критичних ризиків, значущих у контексті резильєнтності. Поінформована готовність, що передбачає планування на випадок непередбачуваних ситуацій, включно з планами мітигації та уникнення загроз, а також реагування на виявлення вразливостей або порушення ланцюга постачання. Розвідка нових типів загроз надає важливу інформацію для інформованої готовності

- *Витримати*. Ця ціль акцентується на створенні механізмів забезпечення гарантованої стійкості критичних функцій системи до негативних впливів і здатності зберігати функціональність навіть в умовах серйозних порушень.

- *Відновити*. Вдосконалення процесу відновлення системи після виникнення загрози або інциденту, включно з поверненням до нормального функціонування.

- *Адаптуватися*. На цьому ключовому, в контексті резильєнтності, етапі створюються можливості для швидкої адаптації системи до змін і нових загроз, оновлюються та модифікуються захисні механізми для підвищення загальної стійкості та ефективності всієї системи.

Слід підкреслити, що принципи, методології та стратегії, що застосовуються для досягнення цілей резильєнтності, значно відрізняються від прийнятих у традиційній практиці інформаційної безпеки. Відмітна особливість резильєнтного підходу проявляється вже на початковому етапі побудови резильєнтної системи, де потрібне проведення комплексної та глибокої трансформації методик ризик-менеджменту та переосмислення теоретичних основ виокремлення та опрацювання ризиків. У рамках цієї трансформації розробляються нові методики, що гармоніюють із концепцією резильєнтності та орієнтовані на ідентифікацію ризиків, які мають ключове значення для стабільності та безперервності функціонування систем або організацій. Метою такого підходу є гарантування постійної працездатності критично важливих функцій в умовах непередбачуваності та динамічних загроз. Це передбачає не просто адаптацію наявних інструментів управління ризиками, а й створення нових, гнучкіших і ефективніших підходів, здатних адекватно реагувати на унікальні виклики, характерні для сучасних інформаційних систем у світі, що безперервно змінюється.

Висновки

У статті викладено результати розробки концептуальної рамки, яка описує особливості резильєнтної поведінки інформаційних систем. Вона вміщує як етапи попередньої підготовки та активної протидії загрозам (типові для класичних сценаріїв інформаційної безпеки), так і механізми адаптації та відновлення після інцидентів, характерні для суто резильєнтних сценаріїв. Важливим аспектом роботи стало визначення ключових задач і цілей резильєнтності, які прямо корелюють із ключовими процесами нейтралізації потенційних загроз, що мають значний вплив на стратегію управління ризиками та підкреслюють необхідність всебічного аналізу критично значущих функцій у контексті організацій.

Особливу увагу приділено розробленню та порівнянню моделей резильєнтних і нерезильєнтних сценаріїв поведінки

систем, а також аналізу відмінностей їхніх реакцій на загрози. Застосування дельта-функції сприяло детальному опрацюванню фаз резильєнтної відповіді на загрози, включно з етапами адаптації та подальшого відновлення.

Література

1. NIST Special Publication 800-160, Volume 2. Developing cyber-resilient systems: A systems security engineering approach. NIST, 2021. 254 p. URL: ttps://doi.org/ 10.6028/NIST.SP.800-160v2r1
2. Korobeynikov F., Bakalynskyi O. Defining the Sequence of Integrating Trustworthiness Components Into Information Security Systems. Ukrainian Information Security Research Journal. 2023. Vol. 4, no. 25. P. 268–274. URL: https://jrnl.nau.edu.ua/ index.php/ZI/article/view/18233.
3. Khuri, A. (2004). Applications of Dirac's delta function in statistics. International Journal of Mathematical Education in Science and Technology, 35, 185 - 195. https://doi.org/10.1080/00207390310001638313.
4. Klau, G., & Weiskircher, R., 2004. Robustness and Resilience. , pp. 417-437. https://doi.org/10.1007/978-3-540-31955-9_15.
5. Benzerga, A., Leblond, J., Needleman, A., & Tvergaard, V. (2016). Ductile failure modeling. International Journal of Fracture, 201, 29-80. https://doi.org/10.1007/s10704-016-0142-6.
6. Venetis, J. 2021. An Explicit Form of Heaviside Step Function. . https://doi.org/10.20944/PREPRINTS202106.0132.V1.
7. Bakalynskyi O., Korobeynikov F. Establishing Goals in the Creation of Cyber-Resilient Systems per NIST. 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 13–15 October 2023. 2023. URL: https://doi.org/10.1109/dessert61349.2023.10416540

Одержано: 11.03.2024

Про автора:

Коробейніков Федір Олександрович,
Аспірант Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова,
Київ, Україна
e-mail: admin@cybersecurity.com.ua
https://orcid.org/0009–0003–8127–4379

ПРОБЛЕМИ ПРОГРАМУВАННЯ
2024 – № 1

УДК 004.424

Вбудування сімейства логічних мов із можливостями перепрограмування монадичної уніфікації в Scala / Р.С. Шевченко, А.Ю. Дорошенко, О.А. Яценко.

У статті запропонована структура для вбудовування методів логічного програмування та програмування в обмеженнях у мову Scala шляхом побудови логічної предметно-орієнтованої мови навколо уніфікації типізованої логіки на основі монад. Розглядаються два види API — високорівневе та низькорівневе, що забезпечує двонаправлений обмін даних під час виконання логічних програм. Відмінності в можливостях логічних механізмів можна виразити як підкласи монади уніфікації. Такий спосіб дає змогу генерувати одну реалізацію налаштовуваної уніфікації для вбудовування різних логічних систем у Scala та використовувати вбудовані сторонні проблемно-орієнтовані мови у логічних виразах. Монадичний прикладний програмний інтерфейс надає розробнику програми простий та інтуїтивно зрозумілий інструмент для реалізації власної логіки всередині уніфікації.

Ключові слова: вбудовування, декларативне програмування, логічне програмування, монада, предметно-орієнтована мова, терм, уніфікація, Scala.

UDC 004.424

Embedding a family of logic languages with custom monadic unification in Scala / R.S. Shevchenko, A.Yu. Doroshenko, O.A. Yatsenko.

The paper proposes a framework for embedding logic programming and constraint programming methods in Scala by building a logical object-oriented language around the unification of typed logic based on monads. Two types of API are considered — high-level for language embeddings and low-level for organization of the bidirectional flow of data during the execution of logic programs. Differences in the capabilities of logical mechanisms can be expressed as subclasses of the class of unification monad types. This design makes it possible to share the implementation of custom unification between different frameworks and to use other languages' embeddings in Scala from the declarative side. The monadic API provides the application developer with a simple and intuitive tool to implement custom logic within the unification. Our frameworks provide a clear representation of logical deduction: Scala code is only used for ad hoc unification. But the overall goal execution is an external interpretation that can implement different strategies. This design provides modularity and good integration with the rest of the ecosystem.

Keywords: declarative programming, embedding, logical programming, monad, object-oriented language, term, unification, Scala.

УДК 004.4

Модифікований метод пошуку ключових слів та термінів у текстових даних / Д.О. Бухаленков, Т.М. Заболотня

У даній статті розглядається питання автоматизованого пошуку ключових слів та термінів у текстових даних. Для підвищення ефективності засобів автоматизованого пошуку ключових слів у тексті за критеріями абсолютної точності та повноти за Жаккаром розроблено модифікацію одного з найсучасніших методів для пошуку ключових слів. Запропоновано модифікацію існуючого гібридного методу пошуку ключових слів, що враховує складні залежності між парами слів у тексті для визначення багатослівних виразів, що, на відміну від оригінального методу, дозволяє знаходити ключові терміни, які складаються з кількох слів. Проведені випробування створеної модифікації гібридного методу пошуку ключових термінів показали ефективність її використання для пошуку ключових термінів у текстах у порівнянні з існуючими аналогами.

Ключові слова: ключові слова, ключові терміни, оброблення текстових даних, Python, стенфордська класифікація.

UDC 004.4

Modified method of searching keywords and keyterms in text data / D.O. Bukhalenkov, T.M. Zabolotnia

This article discusses the issue of automated search for keywords and key terms in text data. To improve the efficiency of the tools of automated search for keywords in the text according to the criteria of absolute accuracy and Jaccard index, a modification of one of the most modern methods for searching for keywords has been developed. A modification of the existing hybrid keyword search method is proposed. It takes into account complex dependencies between pairs of words in the text to determine multi-word expressions, which, unlike the original method, allows finding key terms consisting of several words. Tests of the created modification of the hybrid method of searching for key terms showed the effectiveness of its use for searching for key terms in texts in comparison with existing analogues.

Keywords: keywords, key terms, text data processing, Python, Stanford classification.

УДК 004.93’1

Розумний чат-бот для оцінки емоційного забарвлення повідомлення та відповідної відповіді / В.Р. Кобченко, В.М. Шимкович, А.О. Новацький, П.І. Кравець, Л.Л. Шимкович, А.Ю. Дорошенко

Створено рекурентну модель нейронної мережі, базу даних, призначену для навчання нейронної мережі, програмний інструмент, що її реалізує, для взаємодії з ботом. Для успішного навчання та перевірки моделі було зібрано та анотовано великий набір даних, 50 тисяч коментарів, що містять різні відгуки та думки. Його перекладено українською мовою за допомогою автоматичного перекладача. Архітектура моделі нейронної мережі була оптимізована для покращення результатів класифікації. Крім того, була проведена робота над вдосконаленням інтерфейсу користувача. Розроблений додаток протестовано, результати продемонстровано. Отримана модель продемонструвала точність 85% у визначенні настроїв. Реалізована програма має базовий дизайн, який можна налаштувати, а також деякі налаштування для чат-бота. Подальшого покращення якості класифікації моделі можна досягти шляхом збору більшого та краще організованого набору даних або шляхом дослідження інших архітектур RNN.

Ключові слова: рекурентні нейронні мережі, аналіз настроїв, Python, tensorflow, keras, chatbot.

UDC 004.93’1

An intelligent chatbot for evaluating the emotional colouring of a message and responding accordingly / V.R.Kobchenko, V.M. Shymkovysh, P.I. Kravets, A.O. Novatskyi, L.L. Shymkovysh, A.Yu. Doroshenko

A recurrent neural network model, a database designed for neural network training, and a software tool for interacting with a bot have all been created. A large dataset (50 thousand comments) containing different reviews and sentiments was collected and annotated to successfully train and validate the model. It was also translated into Ukrainian language with the help of an automatic translator. The architecture of the neural network model underwent optimization to enhance classification outcomes. Furthermore, work was conducted on enhancing the user interface. The developed application was tested, and the results were demonstrated. The resulting model demonstrated accuracy 85% in determining sentiments. The implemented application has got basic design (which can be customized) and some settings for chatbot. Further improvement of the model’s classification quality can be achieved by collecting a larger and better organised dataset or by researching other RNN architectures.

Keywords: recurrent neural networks, sentiment analysis, Python, tensorflow, keras, chatbot.

UDC 517.958:57 +519.711.3 + 612.51.001

Problems associated with creating special software for simulating of human physiological responces to dynamic $\pm G_z$ accelerations / R.D. Grygoryan

Under extreme accelerations, human physiological mechanisms cannot provide adequate circulation. Special methods and devices protecting pilot's brain and eye functionality have been proposed but their efficiency is individual and depends on pilot's skills. Currently, the lonely technology to safely acquire and test the necessary skills is based on use of special centrifuges. However, lack of adequate data about physiological and biomechanical events are two main causes worsening the training results. Special computer simulators, capable to model and visualize the main mechanical and physiological effects occurring under dynamic accelerations, could increase the effectiveness of future pilot's training process. This publication aims to define fundamental problems concerned with creating the required software. There exist two main groups of problems. The first group is concerned with the necessity to create basic mathematical models quantitatively describing both the physiological events and effects induced by protective maneuvers. Here special logical procedures, individualizing the basic physiological models, have to be proposed. The second group of problems is predominantly technical and associated with the necessity of special user interface (SUI) development. SUI must be subdivided into two functional sections – one for preparing a single computer experiment (simulation), and another – for analyzing the results of simulation. An experiment preparation includes the following events: i) a preliminary tuning of models according to biometrical data; ii) a setting of acceleration profile; iii) a choosing of protective algorithms and tools (or without protections); iv) a choosing of forms for results storage. Graphs presenting the dynamics of input and output variables are the main forms while the table forms are also included. The user (trainer or trainee) will be able to retrieve from the memory graphs of previous simulations to compare the effectiveness of additional protective elements. The software must be autonomic for the Windows platform.

Keywords: human extreme physiology, quantitative models, simulator, training, information technology.

УДК 517.958:57 +519.711.3 + 612.51.001

Проблеми створення спеціального програмного забезпечення для моделювання фізіологічних реакцій людини на динамічні $\pm G_z$ прискорення / Р.Д. Григорян

Під час екстремальних прискореннях фізіологічні механізми людини не можуть забезпечити належний кровообіг. Були запропоновані методи та пристрої для захисту мозку та очей пілота. Їхня ефективність індивідуальна і залежить від навичок пілота. Наразі єдина технологія безпечного отримання та перевірки необхідних навичок базується на використанні спеціальних центрифуг. Однак відсутність адекватних даних про фізіологічні та біомеханічні події є двома основними причинами погіршення результатів тренувань. Підвищити ефективність процесу підготовки майбутнього пілота могли б спеціальні комп'ютерні тренажери, здатні моделювати та візуалізувати основні механічні та фізіологічні ефекти, що виникають у випадку динамічних прискорень. Ця публікація має на меті визначити фундаментальні проблеми, пов'язані зі створенням необхідного програмного забезпечення. Існує дві основні групи проблем. Перша група пов'язана зі створенням базових математичних моделей, які кількісно описують фізіологічні події та ефекти, спричинені захисними маневрами. Тут повинні бути запропоновані спеціальні логічні процедури, що індивідуалізують основні фізіологічні моделі. Друга група проблем має переважно технічний характер і пов'язана з необхідністю розробки спеціального інтерфейсу користувача (СІК). СІК необхідно розділити на дві функціональні частини – першу для підготовки одного комп'ютерного експерименту (моделювання), а другу – для аналізу результатів моделювання. Підготовка експерименту включає наступні заходи: і) попереднє налаштування моделей за біометричними даними; іі) налаштування профілю прискорення; ііі) вибір захисних алгоритмів та інструментів (або без захисту); іv) вибір форм для зберігання результатів. Графіки, що представляють динаміку вхідних і вихідних змінних, є основними формами, а табличні форми також включені. Користувач (тренер або стажер) зможе отримати з пам'яті графіки попередніх симуляцій для порівняння ефективності додаткових захисних елементів. Програмне забезпечення має бути автономним для платформи Windows.

Ключові слова: екстремальна фізіологія людини, кількісні моделі, тренажер, навчання, інформаційна технологія.

УДК 621.09.04

Програмування одновимірного та двовимірних токенів для токенізації земельних ділянок / С.В. Струтинський, В.А. Яланецький

У публікації розглянуто ключові аспекти застосування блокчейн-інструментів, що дозволяють оперувати частинами віртуальних об'єктів. Для вирішення практичних задач пропонується використовувати одновимірні та двовимірні токени. Розроблено алгоритми та програмно реалізовано одновимірні токени на базі існуючих платформ смарт-контрактів. Запропоновані рішення дозволяють розділяти одновимірні об'єктів на частини та проводити транзакції із цими фрагментами. Розроблено алгоритми реалізації двовимірних токенів та розглянуті особливості їх застосування для представлення земельних ділянок та віртуальних територій. Визначено основні переваги двовимірних токенів у порівнянні із NFT при їх використанні для представлення ділянок земної поверхні. Розроблено алгоритми, що забезпечують володіння віртуальними об'єктами на різних рівнях.

Ключові слова: блокчейн, EVM, смарт-контракт, NFT, фракційні токени

UDC 621.09.04

Programming of one-dimansional and two-dimensional tokens for tokenization of land plots / S. V. Strutynskyi, V. A. Yalanetskyi

The use of blockchain tools that allows spliting virtual objects into parts is considered. Examples of practical use of the developed algorithms are presented. The concept of one-dimensional and two-dimensional tokens representing one-dimensional and flat objects is proposed. Algorithms for the implementation of one-dimensional tokens are developed, and the peculiarities of their practical application are considered. A designed smart contract allows to conduct a basic list of operations with one-dimensional tokens. Algorithms, providing implementation of two-dimensional tokens, are proposed. Peculiarities of presenting territories of virtual worlds and land plots are suggested. A comparative analysis of the use of NFT and two-dimensional tokens for presenting the Earth surface areas is performed. Methods that ensure ownership of tokens at different levels are proposed.

Keywords: Blockchain, EVM, smart contracts, NFT, fractional tokens

УДК 681.3

Класифікація низькочастотних сигналів за допомогою методів кластеризації / Д.В. Рагозін, А.Ю. Дорошенко

У статті розглянуто методи класифікації сигналів звукового та інфразвукового діапазону за допомогою алгоритмів кластеризації у випадках, якщо присутня лише загальна апріорна інформація, наприклад, невідомі типи об'єктів, які генерують відповідні сигнали. Розглянуто підготовку даних звукового діапазону, особливості первинної обробки набору даних, параметри вибору алгоритму залежно від особливостей набору даних. Подано приклади кластеризації набору даних за допомогою алгоритму OPTICS, описано можливості каскадної обробки набору даних.

Ключові слова: класифікація даних, навчання без контролю, кластеризація даних, обробка звуку.

UDC 681.3

Low frequency signal classification using clustering methods / D.V. Rahozin, A.Yu. Doroshenko

The article considers the problem of low frequency signal classification, as sound or vibration pattern footprints may describe types of objects very well. In cases of a priori absence of object signal pattern information, the unsupervised learning methods based on clustering looks good enough for classification, and outperform neural net-based methods in case of limited power envelope. We have used big real-world sound and vibration data set to check several clustering methods (K-Means, OPTICS) for classification without any a priori data and have got good enough results. The article considers data set preparations including primary signal processing and the parameters to select appropriate clustering algorithms, which depends on input data shape. There are several examples of data classification, also cascaded methods for data set improvement are considered. Finally, we provide a good and practical guide for exploring low frequency signals using clustering methods, which can be used for real world observations and analysis for open space and inside buildings.

Keywords: data classification, unsupervised learning, data clustering, sound processing.

УДК 51:681.3.06

Автоматизація глибокого навчання на прикладі уточнення чисельних метеорологічних прогнозів / А.Ю. Дорошенко, Р.В. Кушніренко

Зроблено короткий огляд застосування “глибокого навчання” до науково-технічних задач. Описані проблеми, що можуть виникнути у разі цих застосувань. Показано важливість автоматизації розробки методів “глибокого навчання”. Перевірено можливість застосування нейроеволюційного підходу до проєктування моделей “глибокого навчання”, призначених для постпроцесингу результатів метеорологічного прогнозування (на прикладі приземної температури), отриманого за допомогою чисельних гідродинамічних методів. Показано, що в половині випадків і значення кореня середнього квадратичного відхилення, і відсоток покращених прогнозів для нейроеволюційного підходу є кращими (а в окремих випадках набагато кращими) за відповідні значення для підібраної вручну архітектури. Таким чином, на прикладі задачі уточнення чисельних метеорологічних прогнозів показано, що нейромережеві моделі, отримані автоматично, можуть перевершувати моделі, спроєктовані вручну.

Ключові слова: “глибоке навчання”, автоматизація проєктування нейромереж, нейроеволюція, метеорологічне прогнозування.

UDC 51:681.3.06

Automatic development of deep neural networks for improving numerical meteorological forecasts / A.Yu. Doroshenko, R.V. Kushnirenko

This paper briefly describes the examples of deep learning applications to scientific and technical problems, as well as the difficulties that may arise with these applications. The paper shows the importance of the automatic development of deep neural networks. The paper verifies the possibility of the application of the neuroevolutionary approach to the post-processing of the results of meteorological forecasting (2m temperature) obtained using numerical hydrodynamic methods. The results show that in half of the cases, both the root-mean-square error value and the percentage of improved predictions are better (and in some cases much better) for the neuroevolutionary approach than the corresponding values for the manually designed architecture. Thus, neural network models obtained automatically can outperform manually designed models while applied to improving numerical meteorological forecasts.

Key words: deep learning, automatic development of neural networks, neuroevolution, meteorological forecasting.

УДК 004.41

Розробка методології імплементації транзакцій в розподілених системах з мікросервісною архітектурою / А. М. Глибовець, Т. А. Чернова, М. М. Глибовець

У роботі описано аналіз проблематики використання мікросервісної архітектури в розподілених системах. Наголос зроблено на гнучкості у виборі технологій, масштабованості та організації команд, які працюють над заданими мікросервісами, технічних і доменних проблемах реалізації транзакцій у порівнянні з монолітною системою. Основну увагу приділено транзакціям, оскільки вони забезпечують дотримання атомарності, консистентності, ізольованості та стійкості над декількома сервісами.

В процесі аналізу сучасних підходів та рішень для роботи з транзакціями в розподілених системах, було виявлено, що одним з ефективних рішень є використання патерну Transactional Outbox. Представлено його реалізацію у вигляді Spring starter. Останній додається до системи, конфігурується та полегшує використання транзакцій і публікацію подій, які є частинами транзакції в мікросервісній архітектурі.

Детально описано створену методологію реалізації розподілених транзакцій на базі черг повідомлень, з використанням вищезазначеного стартера. Визначено базові конфігурації та налаштування черг повідомлень для коректної роботи транзакцій в розподілених системах.

Ключові слова: розподілена система, розподілені транзакції, мікросервісна архітектура, патерн Transactional Outbox, асинхронне спілкування, Kafka, Debezium.

UDC 004.41

Development of a methodology for the implementation of transactions in distributed systems with microservice architecture / A.M. Hlybovets, M.M. Glybovets, T.A. Chernova

The paper describes the analysis of the problems of using microservice architecture in distributed systems. Emphasis is placed on flexibility in the choice of technologies, scalability and organization of teams working on given microservices, technical and domain problems of transaction implementation in comparison with a monolithic system. The main focus is on transactions, as they ensure atomicity, consistency, isolation, and persistence across multiple services.

In the process of analyzing modern approaches and solutions for working with transactions in distributed systems, it was found that one of the effective solutions is the use of the Transactional Outbox pattern. Its implementation in the form of Spring starter is presented. The latter is added to the system, configured and facilitates the use of transactions and the publication of events that are part of a transaction in a microservice architecture.

The developed methodology for implementing distributed transactions based on message queues, using the above-mentioned starter, is described in detail. The basic configurations and settings of message queues for the correct operation of transactions in distributed systems are defined.

Keywords: distributed system, distributed transactions, microservice architecture, Transactional Outbox pattern, asynchronous communication, Kafka, Debezium.

УДК 001.2

Ідентифікація рівня спорідненості наукових спеціальностей на основі даних системи Dimensions / С. Д. Штовба, М. В. Петричко

Ідентифіковано рівні спорідненості наукових спеціальностей у межах Австралійсько–Новозеландської стандартної класифікації наук ANZCRC-2020. Ідентифікація здійснена з використанням інформаційної системи Dimensions шляхом аналізу 33.8 млн публікацій за 2019–2023 рр. Рівень спорідненості оцінено за індексом Жаккара. Встановлено, що із 14535 можливих пар спеціальностей, лише 131 пара має значиму спорідненість з індексом Жаккара, що перевищує 0.05. З них для 20 пар спеціальностей рівень спорідненості є високим, а для 61 пари – середнім.

Ключові слова: ідентифікація, класифікація наук, спорідненість спеціальностей, аналіз даних, індекс Жаккара, наукові публікації, підбір рецензентів, наукометрія, Dimensions, ANZSRC-2020.

UDC 001.2

Research specialties' kinship level identification based on data from Dimensions / S. D. Shtovba, M. V. Petrychko

Knowledge about research specialties' kinship level is needed for solving such problems as: improving current research classification system; detecting similar scientific and educational institutions to set up cooperative relations or perform their reorganization; automatic reviewer assignment for peer reviewing PhD-thesis, papers, grant proposals etc. In this paper research specialties' kinship level is identified according to Australian and New Zealand standard research classification ANZCRC-2020. The identification is done using information system Dimensions by analyzing 33.8 million publications for 2019-2023. The level of kinship is assessed by Jaccard index as the ratio of two specialties common publications' number to the total number of publications in these specialties. It is found, that from 14535 possible pairs of specialties only 131 pairs have significant kinship with Jaccard index greater than 0.05. For 20 pairs among them the kinship level is high, and for 61 pairs – average.

Keywords: identification, research classification, specialties' kinship, data analysis, Jaccard index, research publications, reviewer assignment, scientometrics, Dimensions, ANZSRC-2020.

УДК 004.4`2

Архітектурний каркас бібліотеки для уніфікації взаємодії із блокчейн-мережами / Б.І. Добрянський, І.В.Стеценко

У цьому дослідженні розглядаються проблеми, пов'язані з фрагментованим характером інструментів взаємодії з блокчейн мережами. Робота представляє комплексне дослідження вимог до необхідної уніфікованій архітектурі. У відповідь на зростаючу різноманітність блокчейн-мереж пропонується рішення у вигляді уніфікованої бібліотеки на основі універсального та сумісного інтерфейсу, який оптимізує взаємодію між різними блокчейнами. Описано та досліджено поточний стан інструментів для взаємодії з блокчейн-мережами, проведено огляд існуючих рішень, формулювання принципів проєктування та функцій, що лежать в основі пропонованої уніфікованої бібліотеки. Дослідження спрямоване на підвищення доступності та ефективності взаємодії із блокчейн-мережами, зменшуючи складність, пов'язану з розрізненими інструментами та заохочуючи подальшу співпрацю та розвиток нових підходів в рамках спільноти блокчейн розробників.

Ключові слова: розробка блокчейну, сумісність, децентралізовані додатки, засоби розробки, блокчейн-мережі, комплекти розробки програмного забезпечення (SDK), архітектура програмного забезпечення.

UDC 004.4`2

Architectural framework for a unified blockchain interaction library / B.I. Dobrianskyi, I.V. Stetsenko

This research addresses the challenges posed by the fragmented nature of blockchain development tools, presenting a comprehensive exploration of the imperative need for a unified architecture. In response to the growing diversity of blockchain networks, a solution in the form of a unified library based on a versatile and interoperable interface that streamlines interactions across various blockchains is proposed. The current state of blockchain development tools, an overview of existing solutions, formulation of design principles, and functions underlying the proposed unified library are provided. By mitigating the complexities associated with disparate tools, the research aims to enhance the accessibility and efficiency of blockchain development, encouraging collaboration and innovation within the blockchain community.

Keywords: Blockchain development, Interoperability, Decentralized applications, Development tools, Blockchain networks, Software development kits (SDKs), Software Architecture

УДК 004.056

Концептуальна рамка резильєнтної поведінки інформаційних систем / Ф.О. Коробейніков

Пропоноване дослідження присвячене розробленню концептуальної рамки для опису резильєнтної поведінки інформаційних систем, яка визначає основні операційні фази реакції систем у контексті протидії реалізованим ризикам в момент інциденту. Фази включають у себе як заходи попередньої підготовки та безпосередньої протидії загрозам, так і механізми відновлення та адаптації після інцидентів. Досягнення зазначеної мети передбачає виявлення основних цілей і задач резильєнтності, які безпосередньо пов'язані з процесами нейтралізації потенційних загроз і впливають на стратегію управління ризиками, і зумовлюють необхідність всебічного аналізу критично значущих функцій організацій. У процесі дослідження особливу увагу було приділено створенню моделей резильєнтних і нерезильєнтних сценаріїв реакцій на інцидент та виокремленню поведінкових відмінностей. Застосування дельта-функції дало змогу деталізувати фазу резильєнтної відповіді на загрози, включно з етапами адаптації та подальшого відновлення. Дослідження робить значний внесок у сферу інформаційної безпеки, пропонуючи багаторівневий підхід до підвищення резильєнтності інформаційних систем.

Ключові слова: резильєнтність, захист інформації, інформаційна безпека

UDC 004.056

Developing a conceptual framework for resilience in information systems / F. Korobeynikov

This study focuses on the development of a conceptual framework that delineates the resilience of information systems through the analysis of their key operational stages in the context of countering prevalent threats. These stages encompass preliminary security measures, active counteraction against threats, as well as strategies for adaptation and recovery following security breaches. The realization of this objective necessitates the identification and formulation of fundamental goals and objectives associated with resilience, which play a crucial role in the neutralization of potential threats and significantly influence risk management policies, emphasizing the importance of a comprehensive analysis of the key functions of organizations. Throughout the research, special attention was devoted to the creation of models for resilient and non-resilient behavioral scenarios of systems and their responses to threats. The employment of the delta function facilitated a detailed examination of the stages of resilient response to threats, including the processes of adaptation and subsequent recovery. The work contributes significantly to the field of information security, presenting a multifaceted approach to advancing the resilience of information systems.

Keywords: resilience, data protection, information security

ВИМОГИ ДО ОФОРМЛЕННЯ СТАТЕЙ

Журнал «Проблеми програмування» публікує наукові матеріали, які раніше не публікувалися в інших виданнях.

Мова статті: українська, англійська.* Обсяг статті - від 6 до16 сторінок формату А4. Обсяг матеріалів, що подаються на конференцію УкрПРОГ (кожний парний рік) для публікації у подвійному номері складає від 6 до 8 сторінок формату А4. Автори з України подають матеріали на конференцію українською мовою.

Документ зберігається у форматі doc або docx. Назва файлу включає транслітерацію прізвища автора (авторів), наприклад, “Petrenko.doc”.

Автори можуть користуватися електронною поштою для передачі до редакції тексту статті, ділової переписки та правки при коректурі. E-mail редакції: alengoro@isofts.kiev.ua. Телефон: +380 (96) 418 3082.

1. Оформлення файлу з текстом статті

При підготовці файлу використовуються: стиль нормальний (звичайний) або normal; шрифт Times New Roman, розмір шрифта 12 пт.; міжрядковий інтервал – 1,0; абзацний відступ -1,25 см; вирівнювання – по ширині. У тексті не допускається вирівнювання пропусками; розстановка переносів – автоматична. Формат паперу А4, розміри полів документа – 20 мм. Текст статті після зазначення авторів, назви і анотацій (двома мовами) має бути оформлений **у 2 колонки,** ширина яких – 7,86 см, а пробіл між ними – 1,27 см.

2. Розміщення матеріалу статті

Верхній колонтитул: назва рубрики відповідно до переліку, прийнятому редакцією журналу *(пропонується авторами, остаточно узгоджується редакцією).*

УДК *(зліва під рискою верхнього колонтитулу)**:*** індекс за універсальною десятковою класифікацією; ***DOI:*** в тому ж рядку правіше *(заповнюється редакцією)*.

Автори: ініціали та прізвища авторів, курсив (світлий).

Заголовок 1 **(назва статті):** не містить абревіатур та строго відповідає змісту статті. Шрифт 15 пт, напівжирний, регістр верхній.

Анотація**:** 50-100 слів, не містить абревіатур, не зрозумілих зі змісту статті. Шрифт 10 пт, звичайний.

Ключові слова**:** не більше10 слів, не містить абревіатур, не зрозумілих зі змісту статті, подаються в називному відмінку, розділені комами. Шрифт 10 пт, звичайний.

УВАГА! ***Автори, заголовок статті, анотація і ключові слова*** зазначаються **ДВІЧІ:** українською і англійською мовами. Спочатку мовою статті, потім іншою мовою.

Нижній колонтитул (тільки для першої сторінки) включає стандартну інформацію copy right: перший рядок – прізвища авторів, рік; другий рядок – номер ISSN, назва журналу, рік, номер випуску.

Заголовок 2 **(назва розділу):** шрифт 14 пт, напівжирний; абзац із центральним вирівнюванням, без переносів. Заголовки нижчого рівня (пункти і т.п.) у самостійний абзац не виділяються і проходять першим реченням текстового абзацу, шрифт 12 пт, напівжирний.

Основний текст статті (виступ на конференції) включає такі необхідні елементи:

постановка проблеми в загальному вигляді і її зв’язок з важливими науковими або практичними завданнями;

аналіз останніх досліджень і публікацій, у яких розпочато рішення даної проблеми і на які спираються автори, виділення невирішених раніше частин загальної проблеми, яким присвячується дана стаття;

формулювання цілей статті (постановка задачі виступу на конференції);

виклад основного матеріалу дослідження з повним обґрунтуванням отриманих наукових результатів;

висновки з даного дослідження і перспективи подальших розробок у даному напрямку;

подяка (за наявності такої).

Формули створюються в редакторі Microsoft Equation 3.0 або MathType. Формули, на які є посилання в тексті, повинні мати наскрізну нумерацію. Номер формули друкується в круглих дужках біля краю правого поля. Розмір основного шрифту редактора формул – 12 пт. Розміри символів у формулах: звичайний – 12 пт, великий індекс – 9 пт, дрібний індекс – 7 пт, великий символ – 18 пт, дрібний символ – 11пт. Не допускається масштабування формульних об´єктів.

Рисунки мають бути створені вбудованим редактором Word Picture або експортовані з прикладних програм Windows у графічних форматах (bmp, pcx, gif, jpg або tif). Рисунки розташовуються по центру. Нумерація рисунків здійснюється відповідно до порядку згадування у тексті. Нумеровані підписи розміщуються під рисунком з позначенням «Рис. », далі вказується номер рисунка і текст підпису.

Таблиці мають бути підготовлені стандартним вбудованим в Word інструментарієм "Таблиця". Таблиці нумеруються за порядком згадування. На номер таблиці повинно бути посилання в тексті. Номер таблиці вказується в окремому рядку з вирівнюванням по правій стороні (наприклад, «Таблиця 1»). Назви таблиць розміщуються над таблицею з вирівнюванням по центру. Мінімальний розмір шрифту в таблицях – 11 пт.

Література**:** нумерований список джерел згідно ДСТУ 8302:2015 від 01.07.2016 р., шрифт 11 пт, відступ: спеціальний, навислий, 0,63 см. Джерела з заголовками на латиниці наводяться без перекладу. Інші джерела подаються мовою оригіналу. Приклади оформлення бібліографічних посилань згідно з вимогами ***Harvard Style*** наведені в багатьох публікаціях, наприклад: http://www.staffs.ac.uk/assets/harvard_referencing_examples_tcm44-39847.pdf

Дата надходження статті (матеріалів конференції) позначається цифрами окремим рядком після слова «Одержано:»/"Received:".

Відомості про рецензентів зазначаються редакцією після слова «Рецензенти:».

Дані про авторів**:** мають починатися рядком "Про авторів:", напівжирний курсив. Далі вказуються для кожного з авторів ПІБ повністю, вчений ступінь, наукове звання, посада, обов'язково номер ORCID (сайт ORCID http://orcid.org/).

Дані про місце роботи авторів**:** починаються рядком "Місце роботи авторів:", напівжирний курсив. Далі вказуються місце роботи, адреса, телефон, електронна пошта, контактний телефон особи, відповідальної за зв'язок з редактором.

Для уникнення неузгодженості перелік авторів подається під номерами, що відповідають нумерації закладів, де вони працюють.

3. Оформлення файлу з анотаціями

Файл з анотаціями містить інформацію двома мовами – англійською і українською та має бути оформлений у дві колонки: УДК (шрифт – 8 пт); нижче - назва статті (шрифт – 12 пт, напівжирний); прізвища та ініціали авторів (шрифт – 12 пт); текст анотації, ключові слова (шрифт – 10 пт).

Вимоги до анотації: обсяг від 100 до 250 слів, інформативність, оригінальність, змістовність (відображає основний зміст статті і результати досліджень), структурованість (дотримується логіки опису результатів у статті).

Документ зберігається у форматі doc або docx. Назва файлу подається транслітерацією прізвища автора (авторів), наприклад, "Petrenko_Annot.doc".

*16.07.2020 р. набули чинності положення Закону України «Про забезпечення функціонування української мови як державної». Відповідно до статті 22 «Державна мова у сфері науки» у наукових виданнях не повинно бути вміщено матеріалів іншими мовами, окрім державної, англійської та мов ЄС.

Примітка: Підписний індекс журналу «Проблеми програмування» – **90853**.