НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ISSN 1727-4907

П Р О Б Л Е М И ПРОГРАМУВАННЯ

НАУКОВИЙ ЖУРНАЛ

P R O B L E M S IN PROGRAMMING

SCIENTIFIC JOURNAL

2019

№ 2

Теми випуску:

- ***Інструментальні засоби та середовища програмування***
- ***Моделі та засоби систем баз даних і знань***
- ***Експертні та інтелектуальні інформаційні системи***

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

П Р О Б Л Е М И ПРОГРАМУВАННЯ

науковий журнал

Головний редактор
Андон Пилип Іларіонович
академік НАН України,
директор Інституту програмних систем
НАН України
✉ Інститут програмних систем
НАН України
проспект Академіка Глушкова, 40, корп. 5
03187, Київ-187
☎ Тел.+380 (44) 526 5507
E-mail: andon@isofts.kiev.ua
http://www.pp.isofts.kiev.ua

Адреса для кореспонденції

✉ Інститут програмних систем
НАН України
Проспект Академіка Глушкова, 40
03187, Київ-187

☎ Тел.: +380 (44) 526 5065
Факс: +380 (44) 526 6263
E-mail: iss@isofts.kiev.ua

Затверджено до друку вченою радою Інституту програмних систем НАН України.
Протокол № 3 від 02.05.2019 р.

Редактор *В.П. Замула*
Комп'ютерна верстка *В.П. Замула*

Підписано до друку 06.06.2019. Формат 60x84/8. Папір офс. Ум. друк. арк. 15,11.
Обл.-вид. арк. 12,41. Тираж 120 прим. Ціна договірна. Замовл.

Віддруковано ВД «Академперіодика» НАН України
вул. Терещенківська, 4, м. Київ, 01004

Свідоцтво суб'єкта видавничої справи ДК № 544 від 27.07.2001

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

ПРОБЛЕМИ ПРОГРАМУВАННЯ

науковий журнал

№ 2 квітень – червень 2019

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 IN PROGRAMMING

scientific journal

№ 2 ***April – June*** ***2019***

Founded in March, 1999

CONTENTS

УДК 004.4'24 https://doi.org/10.15407/pp2019.02.003

А.Ю. Дорошенко, Р.В. Кушніренко, О.А. Яценко

ПРОЕКТУВАННЯ ПРОГРАМИ ВІЗУАЛІЗАЦІЇ ЗЕМНОЇ ПОВЕРХНІ З ВИКОРИСТАННЯМ АЛГЕБРО-АЛГОРИТМІЧНОГО ІНСТРУМЕНТАРІЮ

Однією з важливих задач в рамках метеорологічного прогнозування є комп'ютерна візуалізація отриманих результатів прогнозу, зокрема, тривимірне моделювання рельєфу земної поверхні. Одним із найпопулярніших програмних засобів розробки застосунків в області візуалізації даних є OpenGL – специфікація, що визначає незалежний від мови програмування кросплатформний програмний інтерфейс для написання застосунків, що використовують двовимірну й тривимірну графіку. Візуалізація даних є досить складною задачею, тому актуальним є питання створення спеціальних засобів автоматизації, що дозволяли б генерувати програмний код для задач з даної предметної області. У статті виконане налаштування раніше створеного алгебро-алгоритмічного інструментарію на проектування та синтез OpenGL програм. Автоматизоване конструювання програм здійснюється з використанням високорівневої мови, що ґрунтується на системах алгоритмічних алгебр Глушкова. Підхід продемонстровано на проектуванні програми візуалізації рельєфу підстильної поверхні для задачі прогнозування погоди.

Ключові слова: автоматизоване проектування програм, алгебра алгоритмів, візуалізація даних, метеорологічне прогнозування, рельєф земної поверхні, синтез програм, тривимірне моделювання, OpenGL.

Вступ

Системи прогнозування стають все більш актуальними у зв'язку зі впровадженням різноманітних автоматизованих систем прийняття рішень, керування та оцінки ризиків у різних сферах людської діяльності. Зокрема, метеорологічне прогнозування [1], окрім задоволення потреб звичайних користувачів, використовується для розв'язання різноманітних задач у транспортній галузі (наприклад, авіації), сільському господарстві, наукових дослідженнях тощо. Суттєве значення має актуальність отриманого прогнозу, його достовірність та оперативність його отримання. Не менш важливим є наочність відображення отриманих даних.

В роботі [2] спроектовано та реалізовано сервісно-орієнтовану портальну систему для надання послуг метеорологічного прогнозування на мультипроцесорній платформі. В рамках системи виконана двовимірна візуалізація результатів прогнозу (ізоліній) за допомогою OpenGL на графічному прискорювачі [3]. OpenGL [4] є одним із найпопулярніших програмних засобів розробки застосунків у області візуалізації даних. Він визначає незалежний від мови програмування кросплатформний програмний інтерфейс для написання програм, що використовують двовимірну й тривимірну графіку. Реалізація OpenGL підтримується більшістю виробників апаратних і програмних платформ і включає декілька бібліотек. Відмітимо, що візуалізація даних є досить складною задачею, тому актуальним є питання розробки спеціальних засобів автоматизації, що дозволяли б генерувати програмний код для таких задач.

В роботах [5–7] авторами запропоновані теорія, методологія та інструментарій для автоматизованого проектування, синтезу та перетворення програм, що ґрунтуються на засобах алгебр алгоритмів та переписувальних правил. Предметною областю застосування інструментальних засобів, зокрема, є метеорологічне прогнозування.

У даній статті виконане налаштування алгебро-алгоритмічного інструментарію на проектування та генерацію OpenGL програм. Застосування інструментарію продемонстроване на задачі тривимірної візуалізації рельєфу підстильної поверхні. Підстильна поверхня є сукупністю природних і перетворених (змінених та створених людиною) ландшафтів на земній поверхні, що знаходяться у взаємодії з атмосферою в процесі обміну теплом і вологою [1].

1. Інструментарій алгебр алгоритмів та його налаштування на конструювання OpenGL програм

Підхід до проектування програм, що розглядається в даній роботі, ґрунтується на засобах систем алгоритмічних алгебр (САА) В.М. Глушкова [5, 6]. САА призначені для формалізованого подання схем алгоритмів і програм та їх трансформації, зокрема, з метою оптимізації за певними критеріями. САА покладено в основу інструментарію автоматизованої розробки програмного забезпечення [5–7].

1.1. Основні конструкції та інструментарій алгоритмічних алгебр. САА Глушкова (або алгебра Глушкова) є двоосновною алгеброю

$$GA = < \{Pr, Op\};\ \Omega_{GA} >,$$

де Pr й Op – множини логічних умов й операторів, визначених на інформаційній множині IS; Ω_{GA} – сигнатура операцій. Оператори є відображеннями (можливо, частковими) інформаційної множини в себе, логічні умови – предикати на множині IS, які приймають значення тризначної логіки $E_3 = \{0, 1, \mu\}$, де 0 – "хибність", 1 – "істина", μ – "невизначеність". Сигнатура $\Omega_{GA} = \Omega_1 \cup \Omega_2$ складається з системи Ω_1 логічних операцій (диз'юнкції, кон'юнкції, заперечення, лівого множення оператора на умову), що приймають значення в множині Pr, і системи Ω_2 операторних операцій, що приймають значення в множині Op (композиції, альтернативи, циклу та ін.). Операторні конструкції будуть розглянуті далі.

Предикати й оператори поділяються на базисні та складені. Базисні поняття вважаються атомарними, неподільними абстракціями та пов'язані з предметною областю алгоритму, що проектується. Складені елементи будуються з базисних за допомогою операцій послідовного й паралельного виконання операторів та/або логічних операцій.

На САА ґрунтується алгоритмічна мова САА/1 [5, 6], призначена для багаторівневого структурного проектування й документування послідовних та паралельних алгоритмів і програм. Перевагою її використання є можливість опису алгоритмів у природно-лінгвістичній формі. Подання операторів мовою САА/1 називаються САА-схемами.

Основними операторними конструкціями мови САА/1 є такі:

- композиція (послідовне виконання операторів):

"*operator* 1";
"*operator* 2";

- альтернатива (розгалуження):

IF '*condition*' THEN
  "*operator* 1"
ELSE
  "*operator* 2"
END IF;

- цикл:

WHILE '*condition*'
LOOP "*operator*"
END OF LOOP.

В роботах [5–7] розглядаються додаткові операції, призначені для формалізованого проектування паралельних алгоритмів.

Інструментарій проектування та синтезу програм (ІПС) ґрунтується на використанні САА й інтегрує три форми подання алгоритмів: аналітичну, природно-лінгвістичну й графову.

Основними компонентами системи ІПС є такі:

- діалоговий конструктор синтаксично правильних програм (ДСП-конструктор), призначений для проектування схем алгоритмів та синтезу програм цільовими мовами програмування Java, C, C++;
- редактор граф-схем;
- база даних, у якій зберігається опис конструкцій САА і базисних понять, а також шаблони їх програмних реалізацій;

- генератор САА-схем на основі схем більш високого рівня (гіперсхем).

За допомогою ДСП-конструктора виконується порівневе проектування схем алгоритмів зверху вниз за допомогою деталізації мовних конструкцій САА. На кожному кроці конструювання система надає користувачу на вибір лише ті конструкції, підстановка яких у текст схеми, що формується, не порушує її синтаксичну правильність. ДСП-конструктор використовує список конструкцій САА і дерево конструювання алгоритму. У процесі проектування обрані користувачем операції відображаються в дереві з подальшою деталізацією їхніх змінних. Залежно від типу обраної змінної (логічної або операторної), система пропонує відповідний список операцій САА. На основі побудованого дерева виконується генерація схеми алгоритму та тексту програми цільовою мовою програмування.

З метою налаштування системи ІПС на проектування й синтез OpenGL програм, в базу даних системи включені нові конструкції, що розглядаються у наступному підрозділі.

1.2. Розширення САА конструкціями, орієнтованими на проектування OpenGL програм. Основним принципом роботи OpenGL [4] є отримання наборів векторних графічних примітивів у вигляді точок, ліній та багатокутників з подальшою математичною обробкою отриманих даних та побудовою растрової картинки на екрані та/або в пам'яті. Векторні трансформації та растеризація виконуються графічним конвеєром, який можна розглядати як дискретний автомат. Команди OpenGL потрапляють в одну з двох груп: вони або додають графічні примітиви на вхід конвеєра, або конфігурують конвеєр на різне виконання трансформацій.

Основна бібліотека OpenGL обробляє і відображує у буфері кадру графічні примітиви з урахуванням певного числа вибраних режимів. Кожний примітив – це точка, відрізок, багатокутник і т. д. Кожний режим може бути змінений незалежно від інших. Визначення примітивів, вибір режимів та інші операції описуються за допомогою команд у формі викликів функцій прикладної бібліотеки. Примітиви визначаються набором з однієї або більше вершин. Вершина визначає точку, кінець відрізку або кут багатокутника. З кожною вершиною асоціюються певні дані (координати, колір, нормаль, текстурні координати і т. п.), що називаються атрибутами. У переважній більшості випадків кожна вершина обробляється незалежно від інших.

В рамках OpenGL використовується VBO (Vertex Buffer Objects) – технологія, що дозволяє зберігати координати вершин спільно з їх атрибутами у відеопам'яті [4]. Робота з VBO складається з таких етапів:

1) створення буферів, передача в буфери координат і атрибутів вершин, передача даних у відеопам'ять;

2) активація потрібного буфера, візуалізація (рендерінг) буфера та його деактивація;

3) видалення даних у відеопам'яті.

Існує низка бібліотек, створених над або на додаток до OpenGL. Зокрема, в програмі візуалізації поверхні, що розглядається в даній роботі, використовується GLFW (Graphics Library Framework) [8] – кросплатформна бібліотека з відкритим кодом для створення вікон, контексту OpenGL і керування введенням, а також бібліотека GLM (OpenGL Mathematics) – математика для OpenGL [9].

Далі наведено список основних базисних операторів, призначених для проектування OpenGL програм і доданих в алгебру алгоритмів.

1. Створення контролера вікна:

"Create and initialize window controller (*controller*)",

що включає ініціалізацію бібліотеки GLFW та створення вікна застосунку.

2. Генерація імені масиву атрибутів вершин зі збереженням його у змінній *var*:

"Generate vertex attribute array name (*var*)".

3. Генерація імені буфера:

"Generate buffer name (*var*)".

4. Зв'язування масиву атрибутів вершин з іменем:

"Bind the vertex attribute array to name (*var*)".

5. Зв'язування буферу з атрибутами вершин:

"Bind the buffer (*buffer*) to vertex attributes".

6. Копіювання масиву даних користувача в буфер атрибутів вершин:

"Copy the user data (*data*) to the vertex attributes buffer".

7. Зв'язування буферу з індексами масиву вершин:

"Bind the buffer (*buffer*) to vertex array indices".

8. Копіювання масиву даних користувача в буфер індексів вершин:

"Copy the user data (*data*) to the vertex indices buffer".

9. Зберігання положення атрибута *atr_name* в шейдерній програмі *program* у змінну *var*:

"Save the location of an attribute (*atr_name*) in the shader program (*program*) to variable (*var*)".

10. Активація масиву атрибутів вершин:

"Enable a generic vertex attribute array (*arr*)".

11. Видалення зв'язку масиву атрибутів вершин з іменем:

"Unbind the vertex attribute array to a name".

12. Створення та використання шейдерної програми (коду, що виконується на графічному прискорювачі) включає в себе такі оператори:

"Create shader program (*prog_id*)";
"Create vertex/fragment shader (*var*) based on the source file (*file*)";
"Attach shader (*shader_id*) to program (*prog_id*)";
"Link shader program (*prog_id*)";
"Check shader program (*prog_id*)";
"Use shader program (*prog_id*)",

де *prog_id* – цілочислена змінна, у якій заданий ідентифікатор програми; *file* – файл з вихідним програмним кодом вершинного або фрагментного шейдера. Шейдери під'єднуються до програми *prog_id*, після чого виконується її збирання, перевірка та використання.

13. Установка матриці моделі виду проекції (Model View Projection, MVP) для контролера і шейдерної програми:

"Set MVP matrix for controller (*controller*) and shader program (*program*)".

14. Візуалізація трикутників на основі масиву даних користувача:

"Render triangles from the user array data (*data*)".

15. Видалення масиву атрибутів вершин:

"Delete vertex attribute array (*arr*)".

16. Видалення буферу:

"Delete buffer (*buffer*)".

Приклад використання вищерозглянутих конструкцій наведено у розділі 2.

2. Застосування системи ІПС для проектування програми візуалізації рельєфу

У даному розділі розглядається використання інтегрованого інструментарію для автоматизованої розробки OpenGL

програми візуалізації рельєфу підстильної поверхні.

Далі наведено САА-схему основного алгоритму візуалізації, побудовану за допомогою системи ІПС. Схема містить один складений оператор – main, у якому використовуються операції САА та базисні оператори, розглянуті у попередньому розділі. У схемі, зокрема, вказано такі змінні: *terrain* – масив даних користувача (точки поверхні, задані координатами *x, y, z*), що зчитуються з бінарного файлу *input_bin*; *VAO* – масив атрибутів вершин; *VBO* – буфер атрибутів вершин; *EBO* – буфер індексів вершин.

```
SCHEME TERRAIN SURFACE
VISUALIZATION ====
    "Terrain surface visualization using
     OpenGL"
  END OF COMMENTS

"main"
==== "Create and initialize window
       controller (controller)";
     "Create mesh (terrain) based on the
       binary data from file (input_bin)";
     "Create shader program (program)";
     "Declare the list of variables (VAO,
      VBO, EBO) of type (unsigned int)";
     "Generate vertex attribute array name
      (VAO)";
     "Generate buffer name (VBO)";
     "Generate buffer name (EBO)";
     "Bind the vertex attribute array to name
       (VAO)";
     "Bind the buffer (VBO) to vertex
       attributes";
     "Copy the user data (terrain) to the
       vertex attributes buffer";
     "Bind the buffer (EBO) to vertex
       array indices";
     "Copy the user data (terrain) to the
       vertex indices buffer";
     "Save the location of an attribute
       (aPos) in the shader program
       (program) to variable (vertloc)";
     "Define an array of generic vertex
       attribute data (vertloc) of type
       (GL_FLOAT) and size
       (3 * sizeof(float))";
     "Enable a generic vertex attribute array
      (vertloc)";
     "Unbind the vertex attribute array
       to a name";
     WHILE NOT 'Window controlled by
                (controller) should
                close'
     LOOP
       "Specify clear values for the color
        buffers (0.0f, 0.0f, 0.5f, 1.0f)";
       "Clear colour and depth buffers";
       "Enable depth test";
       "Disable polygon culling";
       "Use shader program (program)";
       "Set MVP matrix for controller
        (controller) and shader program
        (program)";
       "Bind the vertex attribute array to
        name (VAO)";
       "Render triangles from the user array
        data (terrain)";
       "Unbind the vertex attribute array
        to a name"
     END OF LOOP;
     "Delete vertex attribute array (VAO)";
     "Delete buffer (VBO)";
     "Delete buffer (EBO)";
     "Return value (0)";

END OF SCHEME TERRAIN SURFACE
VISUALIZATION
```

На рис. 1 показано копію екрану системи ІПС з побудованою схемою.

Далі наведено САА-схему DATAREADER, що містить визначення основних структур даних задачі (вершина *Vertex*, вектори вершин *Points* та індексів *Indices*, сітка *Mesh*), а також загальне визначення класу *DataReader*, призначеного для зчитування точок поверхні з файлу.

```
SCHEME DATAREADER ====
    "Data reader for the terrain surface
     visualization task (header file)"
  END OF COMMENTS
```

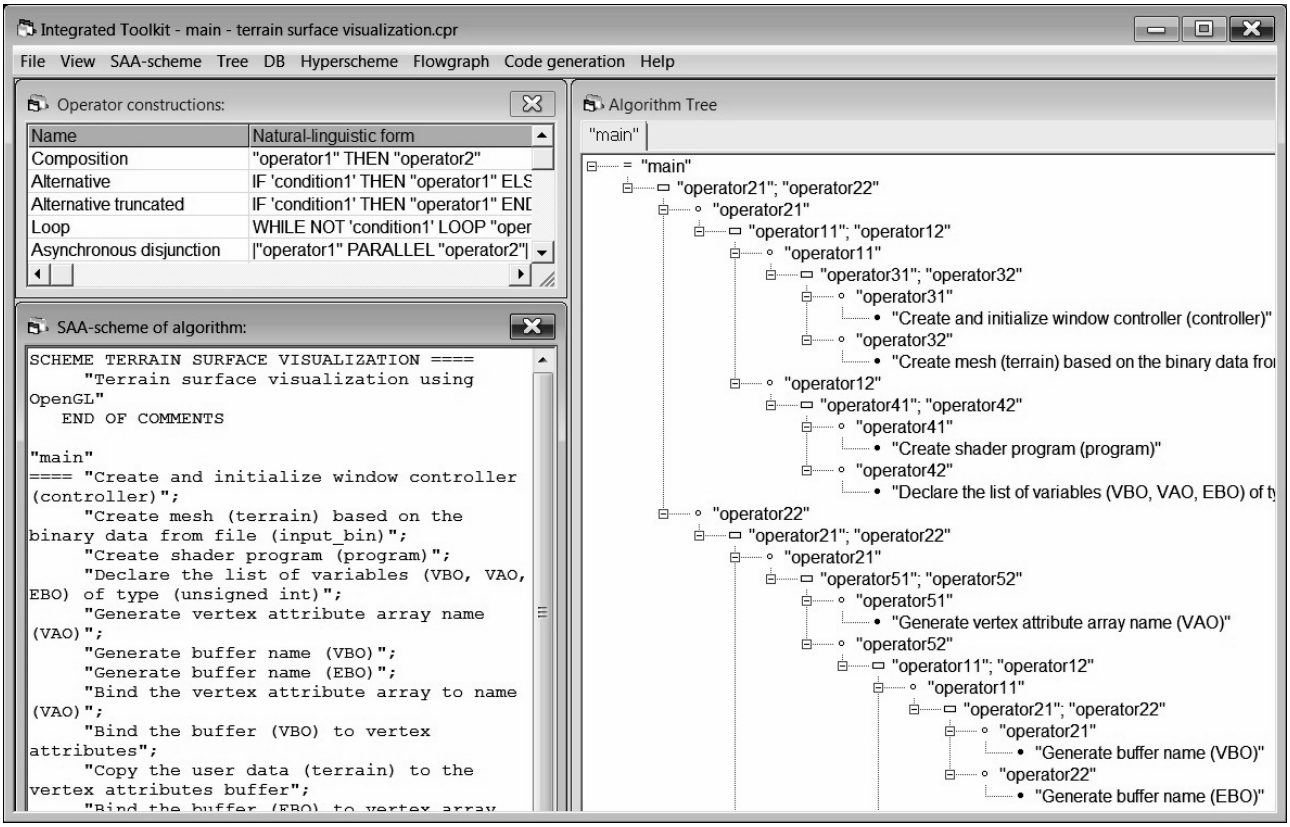


Рис. 1. Копія екрану системи ІПС з побудованою САА-схемою алгоритму візуалізації поверхні

STRUCTURE *Vertex* (
  “Declare a variable (*x*) of type (*float*)”;
  “Declare a variable (*y*) of type (*float*)”;
  “Declare a variable (*z*) of type (*float*)”);

“Define vector type (*Points*) of type (*Vertex*)”;
“Define vector type (*Indices*) of type (*unsigned*)”;

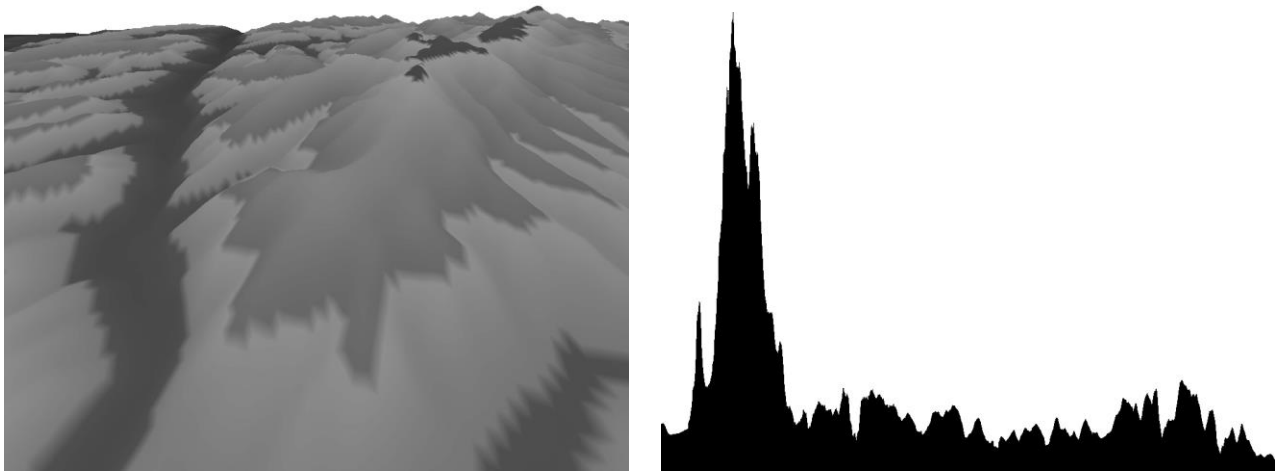
STRUCTURE *Mesh* (
  CONSTRUCTOR *Mesh*(“Declare a constant (*points*) of type (*Points&*)”);
  “Declare a constant (*points*) of type (*Points*)”;
  “Declare a constant (*indices*) of type (*Indices*)”);

CLASS *DataReader*
  PUBLIC:
    CONSTRUCTOR *DataReader*(“Declare a string constant (*filename*)”);
    METHOD *getMesh*(): *Mesh*;
  PRIVATE:
    “Declare a variable (*mesh*) of type (*Mesh*)”
END OF CLASS *DataReader*;

END OF SCHEME DATAREADER

На основі побудованих САА-схем в системі ІПС виконана генерація програмного коду C++, що використовує OpenGL. Окрім реалізації основної функції (main), програма містить чотири основні класи: Controller (контролер вікна застосунку), DataReader (зчитування вхідних даних), Shader та ShaderProgram (шейдерна програма).

Проведено експеримент з виконання розробленої OpenGL програми на графічному прискорювачі. На рис. 2, *а* наведено тривимірне зображення рельєфу підстильної поверхні у межах області моделювання. На рис. 2, *б* показано вертикальний переріз поверхні уздовж 48° пн. ш.



а *б*

Рис. 2. Результати роботи програми візуалізації рельєфу підстильної поверхні:
а – тривимірне зображення рельєфу у межах області моделювання,
б – вертикальний переріз поверхні уздовж 48° пн. ш.

Висновки

Виконане налаштування алгебро-алгоритмічного інструментарію на проектування та синтез OpenGL програм. Автоматизоване конструювання програм виконується з використанням високорівневої мови, що ґрунтується на системах алгоритмічних алгебр Глушкова та орієнтована на природно-лінгвістичну форму подання алгоритмів. В основу інструментарію покладене діалогове проектування схем алгоритмів зверху вниз на основі суперпозиції операцій алгебри алгоритмів. На основі схем виконується генерація коду цільовою мовою програмування. Підхід продемонстровано на проектуванні програми візуалізації рельєфу підстильної поверхні для задачі прогнозування погоди.

Література

1. Прусов В.А., Дорошенко А.Ю. Моделювання природних і техногенних процесів в атмосфері. Київ: Наукова думка, 2006. 542 с.
2. Дорошенко А.Ю., Іваненко П.А., Овдій О.М., Павлючин Т.О., Вітряк Є.А. До створення Інтернет-порталу надання послуг метеорологічного прогнозування на мультипроцесорній платформі. *Проблеми програмування*. 2015. № 3. С. 24–32.
3. Бекетов О.Г., Вітряк Є.А., Мироненко І.О., Овдій О.М. Розвиток Інтернет-порталу метеорологічного прогнозування на мультипроцесорній платформі. *Проблеми програмування*. 2016. № 2–3. С. 246–253.
4. OpenGL – The Industry Standard for High Performance Graphics [Електронний ресурс]. Режим доступу до ресурсу: https://www.opengl.org (дата звернення 19.04.2019 р.).
5. Андон Ф.И., Дорошенко А.Е., Жереб К.А., Шевченко Р.С., Яценко Е.А. Методы алгебраического программирования. Формальные методы разработки параллельных программ. Киев: Наукова думка, 2017. 440 с.
6. Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А. Алгеброалгоритмические модели и методы параллельного программирования. Киев: Академпериодика, 2007. 631 с.
7. Doroshenko A., Zhereb K., Yatsenko O. Developing and optimizing parallel programs with algebra-algorithmic and term rewriting tools. *Communications in Computer and Information Science. Information and Communication Technologies in Education, Research, and Industrial Applications*. Berlin: Springer, 2013. Vol. 412. P. 70–92.
8. GLFW [Електронний ресурс]. Режим доступу до ресурсу: https://www.glfw.org (дата звернення 19.04.2019 р.).
9. GLM. OpenGL Mathematics [Електронний ресурс]. Режим доступу до ресурсу: https://glm.g-truc.net/0.9.9/index.html (дата звернення 19.04.2019 р.).

References

1. Prusov V.A. & Doroshenko A.Yu. (2006) Simulation of natural and anthropogenic processes in the atmosphere. Kyiv: Naukova dumka. (in Ukrainian)
2. Doroshenko A.Yu., Ivanenko P.A., Ovdii O.M., Pavliuchyn T.O. & Vitriak I.A. (2015) Creation of an Internet portal providing meteorological forecasting services on multiprocessor platform. Problems in programming. (3). P. 24–32. (in Ukrainian)
3. Beketov O.G., Vitriak E.A., Myronenko I.O., Ovdii O.M. (2016) Development of meteorological forecasting web portal on multiprocessor platform. Problems in programming. (2–3). P. 246–253. (in Ukrainian)
4. OpenGL – The Industry Standard for High Performance Graphics. [Online] Available from: https://www.opengl.org [Accessed: 19 April 2019]
5. Andon P.I. et al. (2017) Methods of algebraic programming. Formal methods of parallel program development. Kyiv: Naukova dumka. (in Russian)
6. Andon P.I. et al. (2007) Algebra-algorithmic models and methods of parallel programming. Kyiv: Akademperiodyka. (in Russian)
7. Doroshenko A., Zhereb K. & Yatsenko O. (2013) Developing and optimizing parallel programs with algebra-algorithmic and term rewriting tools. In Proc. 9th International Conference "ICT in Education, Research, and Industrial Applications" (ICTERI 2013), Revised Selected Papers. Kherson, Ukraine, 19-22 June 2013. Berlin: Springer. 412. P. 70-92.
8. GLFW. [Online] Available from: https://www.glfw.org [Accessed: 19 April 2019]
9. GLM. OpenGL Mathematics. [Online] Available from: https://glm.g-truc.net/0.9.9/index.html [Accessed: 19 April 2019]

Одержано 25.04.2019

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень,
професор кафедри автоматики та
управління в технічних системах
НТУУ "КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в
українських виданнях – понад 150.
Кількість наукових публікацій в
зарубіжних виданнях – понад 50.
Індекс Хірша – 5.
http://orcid.org/0000-0002-8435-1451,

Кушніренко Роман Владиславович,
магістр кафедри Теорії та технології
програмування факультету комп'ютерних
наук та кібернетики.
https://orcid.org/0000-0002-1990-8727,

Яценко Олена Анатоліївна,
кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість наукових публікацій в
українських виданнях – 44.
Кількість наукових публікацій в
зарубіжних виданнях – 14.
Індекс Хірша – 2.
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 3559.

Київський національний університет
імені Тараса Шевченка,
01601, Київ, вул. Володимирська, 60.

E-mail: doroshenkoanatoliy2@gmail.com,
romashka1996@gmail.com,
oayat@ukr.net

УДК 004.4'24 https://doi.org/10.15407/pp2019.02.011

Т.А. Мамедов, А.Ю. Дорошенко

ЗАСІБ НАЛАШТУВАННЯ ПРОГРАМ НА ПЛАТФОРМІ .NET ЗА ДОПОМОГОЮ ПЕРЕПИСУВАЛЬНИХ ПРАВИЛ

Розроблено програмний засіб для оптимізації обчислень, що дозволяє в автоматизованому режимі здійснити оптимізацію програми шляхом підвищення її швидкодії. Для цього реалізовано спеціальний плагін до системи переписувальних правил TermWare, за допомогою якого система здійснює налаштування програм, написаних на платформі .NET. Плагін використовує аналізатор Roslyn, реалізований генератор термів системи TermWare з вихідного коду програми. Програмний засіб проілюстровано на відомому прикладі клітинного автомату «Гра життя» на різних розмірах площини. Під час експериментів проведені виміри швидкодії програми до та після модифікації вихідного коду. Наведене порівняння результатів роботи методу самоналаштування за допомогою TermWare та інструментарію під назвою Eazfuscator.NET. Експерименти розробленого методу та бібліотеки Eazfuscator.NET проведені на персональному комп'ютері.

Ключові слова: клітинний автомат гра життя, автотюнінг, вимірювання швидкодії.

Вступ

Ручна оптимізація займає немалий час в циклі розробки програмних засобів, тому в останні роки розроблюються різні автоматизації цього процесу, зокрема, шляхом самоналаштування програм на цільову платформу. Концепція такого підходу під назвою автотюнінгу вже давно існує в світі інформаційних технологій і з кожним роком стає все більш важливою, оскільки обсяг коду програмних систем і/або кількість оброблюваної інформації зростає все більше і більше, а швидкодія програм стає більш гострою проблемою [1]. На сьогодні створено досить багато систем, які реалізують концепцію автотюнінгу. Серед них, для прикладу, можна назвати такі системи, як ATLAS [2], AbcLibScript [3], TuningGenie [4].

В даній роботі для цієї мети була використана система переписувальних правил TermWare [5], для якої був реалізований parser/printer для мови C#. Також був використаний інструментарій TuningGenie для вимірювання швидкодії програми на різних ділянках коду. Наведено порівняння різних підходів до концепції автотюнінгу, серед яких є і використання такого засобу автоматизації програм на мові C# як Eazfuscator.NET [6].

Матеріал даної роботи організований наступним чином. В розділі 1 описаний алгоритм, який був розроблений на мові C# для експерименту. В розділі 2 коротко описана реалізація плагіну TermWare для роботи з мовою C#. В третьому розділі описано засоби та реалізацію аналізатора мови C#. В останньому розділі описаний експеримент та результати оптимізації програми за допомогою переписувальних правил та інструментарію Eazfuscator.NET.

1. Алгоритм програми «Гра життя»

«Гра життя» являє собою нескінченну, двомірну ортогональну сітку квадратних клітин, кожна з яких знаходиться в одному з двох можливих станів, живих або мертвих (або заселених і ненаселених відповідно) [7]. Кожна клітина взаємодіє зі своїми вісьмома сусідами, які є клітинами, розташованими горизонтально, вертикально або по діагоналі. На кожному кроці відбуваються наступні переходи:

- будь-яка жива клітина з меншою, ніж двома живими сусідами вмирає, як ніби за ненаселення;
- будь-яка жива клітина з двома або трьома живими сусідами живе на наступному поколінні;
- будь-яка жива клітина з більш ніж трьома живими сусідами вмирає через перенаселення;

- будь-яка мертва клітина з трьома живими сусідами стає живою клітиною, за допомогою відтворення.

В реалізації програми для даної роботи на початку вказуються константи, які визначають розмір площини (сітки) та кількість поколінь. Переходячи на наступне покоління, всі правила застосовуються одночасно до всіх клітин на площині та відбувається дискретний момент «народження» або «смерті».

Псевдокод для визначення всіх сусідів для клітини.

```
BEGIN
NUMBER numOfAliveNeighbors=0, x, y, heigth, width;
ARRAY OF ARRAYS cells
FOR (i = x-1; i < x + 2; i++)
{
  FOR (j = y-1; j < y + 2; j++)
  {
    IF (!((i < 0 || j < 0) || (i >= heigth || j >= width)))
    {
      if (cells[i, j] == true)
      {
        numOfAliveNeighbors++;
      }
    }
  }
}
END
```

Псевдокод основної дії (перехід на наступне покоління) наведений далі.

```
BEGIN
NUMBER heigth, width;
ARRAY OF ARRAYS cells;
FOR (i = 0; i < heigth; i++)
{
  FOR (j = 0; j < width; j++)
  {
    numOfAliveNeighbors=CalcNeigh(i, j);
    IF (cells[i, j] == TRUE)
    {
      IF (numOfAliveNeighbors < 2 AND numOfAliveNeighbors > 3)
      {
        cells[i, j] = FALSE;
      }
    }
    ELSE
    {
      IF (numOfAliveNeighbors == 3)
      {
        cells[i, j] = TRUE;
      }
    }
  }
}
END
```

2. Система Termware

Для здійснення всіх перетворень використовується платформа переписувальних правил Termware, докладно описана в [5]. Termware призначена для опису перетворення над термами, тобто виразами виду $f\ (t_1 \ldots t_n)$. Для завдання перетворень використовуються правила Termware, тобто конструкції виду *source [condition]* –> *destination [action]*.

Тут *source* – вихідний терм (зразок для пошуку), *condition* – умова застосування правила, *destination* – перетворений терм, *action* – додаткове дія при спрацьовуванні правила. Кожен з 4 компонентів правила може містити змінні (які записуються у вигляді $ var), що забезпечує спільність правил. Компоненти *condition* і *action* є необов'язковими. Вони можуть виконувати довільний процедурний код, зокрема використовувати додаткові дані про програму.

Застосування правила відбувається наступним чином: спочатку знаходиться підтерм вхідного терма (дерева програми), який підходить під *source*. Далі перевіряється умова застосування (якщо вона є). Якщо умова виконується, відбувається заміна *source* на *destination*. При цьому змінні в *destination* замінюються відповідними значеннями з *source*. Також виконується дія *action* (якщо воно було присутнє).

Кожне перетворення задається системою правил, тобто набором правил, які послідовно застосовуються до даного тер-

му (дереву програми). Порядок застосування правил визначається стратегією. У систему Termware вбудовані кілька основних стратегій, таких як TopDown, BottomUp, FirstTop. Крім того, можливе створення додаткових стратегій.

3. Реалізація плагіну для TermWare для роботи з мовою C#

3.1. Аналізатори мови C#. Існує досить багато аналізаторів для мови C#, такі як Metaspec C# parser library та інші, описані в роботі [6]. Але проблематика аналізаторів мови C# полягає у тому, що мова має вже вісім версій. У 2015 році вийшла платформа Roslyn [8] з відкритим вихідним кодом, що розробляється корпорацією Microsoft, і містить у собі компілятори і засоби для розбору і аналізу коду, написаного на мовах програмування C # і Visual Basic.

Різні нововведення на зразок code fixes реалізуються саме за рахунок використання Roslyn.

За допомогою засобів аналізу, наданими платформою Roslyn можна проводити повний розбір коду, аналізуючи всі підтримувані конструкції мови.

Середовище Visual Studio дозволяє створювати на основі Roslyn як вбудовані в саму IDE інструменти (розширення Visual Studio), так і незалежні додатки (standalone інструменти).

3.2. Реалізація. В даній роботі був використаний фреймворк TermWare, в якому розроблений концепт термінальних систем об'єктних структур та переписувальних правил. TermWare написаний на мові Java, але використовує окремо плагін Parser/Printer. Тому незважаючи на те, що фреймворк написаний на одній технології, він надає можливість мультиплатформності. Для цього був створений екземпляр TermWareInstance – це, образно говорячи, стрижень, який містить у собі ієрархічну систему доменів і словники імен парсерів мов та принтерів. Для роботи з будь-яким іншим стеком технологій треба реалізувати 2 інтерфейси: IParser, IParserFactory.

Загальний паттерн використання цих інтерфейсів наступний:

- програміст визначає парсер мови, відповідний інтерфейсу IParser і фабрику IParserFactory;
- перед використанням розбирача цієї мови повинна бути зареєстрована своя фабрика парсерів для відповідної мови, використовуючи метод TermWareInstance. addParserFactory (String language, IParserFactory factory).

Після цього виклик loadFile(fname, X) буде використовувати парсер відповідної мови X.

Для синтаксичного аналізу програм було використано можливості відкритого програмного забезпечення – компілятора Roslyn. За допомогою компілятора можна отримати синтаксичне дерево реалізованої програми, яке і потрібно для парсеру.

Оскільки Roslyn працює на CLR, а парсер/принтер повинен працювати на JVM, для розв'язування цієї проблеми був використаний міст jni4net, який дозволяє реєструвати будь-який dll файл, згенерований за допомогою інструменту proxygen, який створює обгортку до будь-якої бібліотеки [9]. Це дозволяє використовувати та створювати екземпляри класів з C# на Java та навпаки з Java на C#.

Також наведена спрощена UML-діаграма класів відповідного парсеру на рис. 1.

Далі буде наведений приклад терму для методу NextGeneration().

```
For(Assignment(i,0), i<=Heigth, Increment(i),
For(Assignment(j,0), i<=Width, Increment(j),
[DeclarationAssignment(numOfAliveNeighb
ors, int, [MethodCall(CalcNeighbor, i, j)],
If(ArrayElement(ArrayElement(cells, i), j), [
        If (numOfAliveNeighbors<2, [
Assignment(ArrayElement(ArrayElement(cell
s, i), j), false)])
        If (numOfAliveNeighbors>3, [
Assignment(ArrayElement(ArrayElement(cell
s, i), j), false)])])
        If(NotEqual(ArrayElement(ArrayEle
ment(cells, i), j)), true),
        [If (Equal(numOfAliveNeighbors, 3)),
[Assignment(ArrayElement(ArrayElement(ce
lls, i), j), true)
]))])])]
```

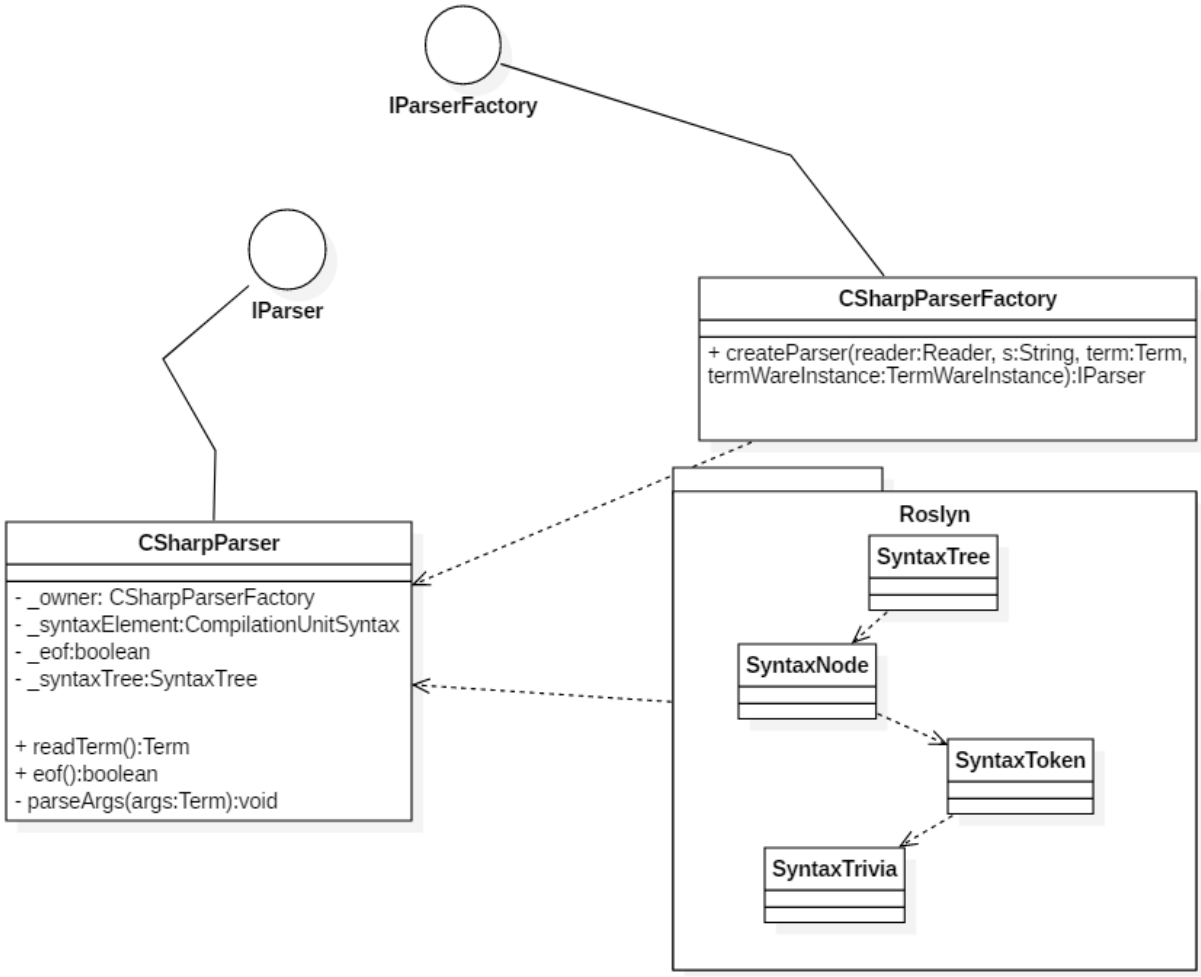


Рис. 1. Спрощена діаграма класів плагіну для TermWare

Наступний код – це приклад терму для методу визначення сусідів клітини CalcNeighbor(x, y).

```
[DeclarationAssignment(numOfAliveNeighb
ors, int, 0),
For(Assignment(i, x – 1), i < x + 2,
Increment(i),
For(Assignment(j, y – 1), j < y + 2,
Increment(j),
    If(Not(Or(Or(i<0,j<0),Or(i>=Heigth,j
>=Width))),
[If(ArrayElement(ArrayElement(cells, i), j),
[Increment(numOfAliveNeighbors)]
)])]
```

4. Експеримент

Для визначення ефективності програмних перетворень проведено експеримент із застосуванням двох видів автоматичної оптимізації програми.

Експеримент складався з порівняння результатів виконання програм, отриманих за допомогою описаних систем.

Експеримент проводився з використанням комп’ютера, оснащеної процесором Intel Core i7-7700HQ 2.80GHz з обсягом оперативної пам'яті в 16 Гб.

Масштабування дослідної задачі регулюється за допомогою вибору розміру площини, на якому здійснюється «Гра життя», а також кількість поколінь, які повинні пройти перед закінченням виконання програми.

Експеримент проводився на різних розмірах площини, починаючи від 300 і закінчуючи розмірами в 1200000 клітин.

Результат експериментів виводиться на Excel файл та показаний на рис. 2, де можна побачити, що застосування системи TermWare є більш ефективним з точки зору швидкодії в порівнянні з бібліотекою Eazfuscator.NET.

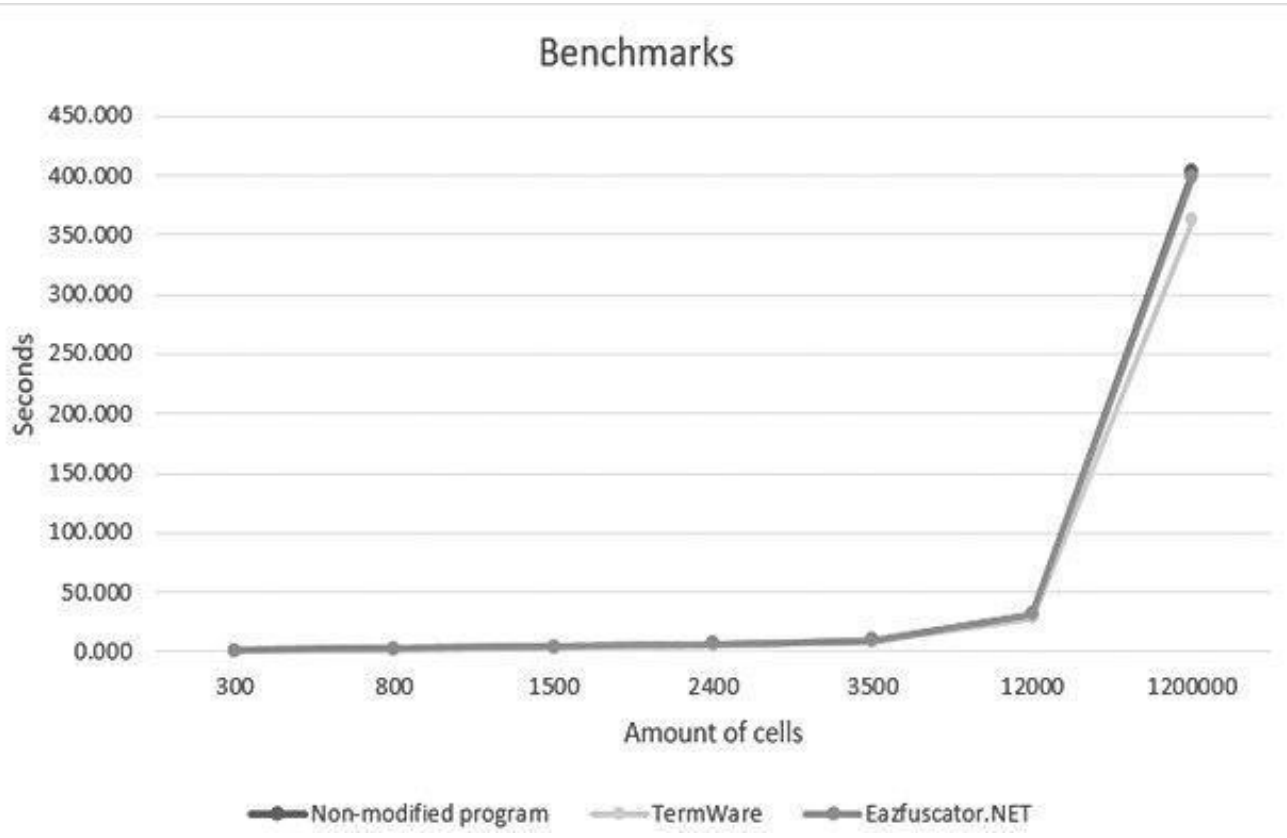


Рис. 2. Результат тестів програми

Як видно з графіка, система TermWare випереджає Eazfuscator.NET, коефіцієнт прискорення порівняно з немодифікованою програмою і TermWare складає приблизно на 1.34, а з бібліотекою Eazfuscator.NET – на 1.15.

З іншого боку, якщо порівнювати довжину коду, то в цьому випадку Eazfuscator.NET не потребує ніяких модифікацій коду для застосування, тому в цьому випадку ця бібліотека краща за використання TermWare.

Висновок

Розроблено та реалізовано плагін системи переписувальних правил TermWare для роботи з мовою C#, що дозволяє оптимізувати програми, написані цією мовою, в атоматизованому режимі у напрямку покращення їх швидкодії. Проведено порівняння з фреймворком Eazfuscator.NET на відомому академічному прикладі клітинного автомата «Гра життя».

Література

1. Naono K., Teranishi K., Cavazos J., Suda R. (2010), Software Automatic Tuning From Concepts to State-of-the-Art Results. – Berlin: Springer.
2. R Clinton Whaley, Jack J Dongarra, ATLAS. Encyclopedia of Parallel Computing. 2011. С. 95–101.
3. Katagiri T., Kise K., Honda H., Yuba T., AbcLibScript: A directive to support specification of an auto-tuning facility for numerical software. Parallel Computing. 2006. С. 92–112.
4. Pavlo A. Ivanenko, Anatoliy Y. Doroshenko, and Kostiantyn A. Zhereb, TuningGenie: Auto-Tuning Framework Based on Rewriting Rules // in: 10th International Conference, ICTERI 2014, Kherson, Ukraine, June 9-12, 2014, Revised Selected Papers, Series: Communications in Computer and Information Science, (Ermolayev, V., Mayr, H.C., Nikitchenko, M., Spivakovsky, A., Zholtkevych, G. (Eds.)), Springer. CCIS Vol. 469. 2014. P. 139–160.
5. TermWare [Електронний ресурс]. Режим доступу до ресурсу: http://www.gradsoft.ua/products/termware_rus.html
6. Eazfuscator.NET [Online]. Available from: https://www.gapotchenko.com/eazfuscator.net

7. Doroshenko A., Shevchenko R. A rewriting framework for rule-based programming dynamic applications. Fundamenta Informaticae. 2006. Vol. 72, N 1–3. P. 95–108.
8. Roslyn [Електронний ресурс]. Режим доступу до ресурсу: https://github.com/dotnet/roslyn
9. Жереб К.А., Программный инструментарий, основанный на правилах для автоматизации разработки приложений на платформе Microsoft .NET. *Управляющие системы и машины.* 2009, №4. С. 51–59.
10. JNI4NET [Електронний ресурс]. – Режим доступу до ресурсу: http://jni4net.com/
11. Application of Rewriting Term System for Source Code Analysis [Електронний ресурс]. Режим доступу до ресурсу: http://www.gradsoft.ua/eng/whitepapers/secr2008/secr2008-1.html

References

1. Naono, K., Teranishi, K., Cavazos, J., Suda, R. (2010), Software Automatic Tuning From Concepts to State-of-the-Art Results. – Berlin: Springer.
2. R Clinton Whaley, Jack J Dongarra, ATLAS. Encyclopedia of Parallel Computing. 2011. P. 95–101.
3. Katagiri T., Kise K., Honda H., Yuba T., AbcLibScript: A directive to support specification of an auto-tuning facility for numerical software. Parallel Computing. 2006. P. 92–112.
4. Pavlo A. Ivanenko, Anatoliy Y. Doroshenko, and Kostiantyn A. Zhereb, TuningGenie: Auto-Tuning Framework Based on Rewriting Rules // in: 10th International Conference, ICTERI 2014, Kherson, Ukraine, June 9-12, 2014, Revised Selected Papers, Series: Communications in Computer and Information Science, (Ermolayev, V., Mayr, H.C., Nikitchenko, M., Spivakovsky, A., Zholtkevych, G. (Eds.)), Springer, CCIS Vol. 469, 2014. P. 139–160.
5. TermWare [Online]. Available from: http://www.gradsoft.ua/products/termware_rus.html
6. Eazfuscator.NET [Електронний ресурс]. – Режим доступу до ресурсу: https://www.gapotchenko.com/eazfuscator.net
7. Doroshenko A., Shevchenko R. A rewriting framework for rule-based programming dynamic applications. Fundamenta Informaticae. – 2006. – Vol. 72, No. 1–3. – P. 95–108.
8. Roslyn [Online]. Available from: https://github.com/dotnet/roslyn
9. Zhereb K., Программный инструментарий, основанный на правилах для автоматизации разработки приложений на платформе Microsoft .NET // Control systems and computers.(4). P. 51–59. (in Russian).
10. JNI4NET [Online]. Available from: http://jni4net.com/
11. Application of Rewriting Term System for Source Code Analysis [Online]. Available from: http://www.gradsoft.ua/eng/whitepapers/secr2008/secr2008-1.html

Одержано 03.05.2019

Про авторів:

Мамедов Турал Алірзайович,
студент 2 курсу магістратури в Київському національному університеті імені Тараса Шевченка.
Кількість наукових публікацій в українських виданнях – 2.
https://orcid.org/0000-0003-3029-5834,

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії комп'ютерних обчислень,
професор кафедри автоматики та управління в технічних системах
НТУУ "КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в українських виданнях – понад 150.
Кількість наукових публікацій в зарубіжних виданнях – понад 50.
Індекс Хірша – 5.
http://orcid.org/0000-0002-8435-1451

Місце роботи авторів:

Київський національний університет імені Тараса Шевченка, кафедра теорії та технології програмування.

Національний технічний університет України "КПІ імені Ігоря Сікорського" та Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
E-mail: tural.mamedov1@gmail.com,
doroshenkoanatoliy2@gmail.com

УДК 004.853, 004.55 https://doi.org/10.15407/pp2019.02.017

Ю.В. Рогушина

ПРОБЛЕМИ ВИКОРИСТАННЯ ОНТОЛОГІЧНОГО АНАЛІЗУ ДЛЯ ПОДАННЯ ЗНАНЬ У WIKI-РЕСУРСАХ

Проаналізовано напрямки застосування Wiki-технологій для створення інформаційних ресурсів великого обсягу та складної структури, орієнтованих на функціювання у відкритому середовищі Web. Обґрунтовано потребу в інтелектуалізації Wiki-технології, а саме – доцільність використання Semantic MediaWiki для отримання необхідної виразної потужності у поданні семантики основних елементів Wiki-ресурсу, їх властивостей та зв'язків між ними. Розглянуто зв'язок між поданням знань у Semantic MediaWiki та стандартами Semantic Web, які використовуються для інтеграції інтелектуальних застосунків та баз знань у Web. Продемонстровано на прикладах доцільність застосування онтологічного аналізу та побудови онтологічної моделі для формального та однозначного відображення структури та контенту бази знань онлайн-версії «Великої української енциклопедії» (e-BYE). Запропоновано методи та засоби застосування цієї онтологічної моделі для створення контенту e-BYE та більш ефективного пошуку та навігації у цьому інформаційному ресурсі.

Ключові слова: онтологія, Semantic Web, Wiki-ресурс, Wiki-онтологія.

Вступ

Збільшення обсягів інформації у Web, поширення Big Data викликають потребу у переході від збереження та накопичення даних до побудови та збереження знань, придатних для повторного використання в інформаційних системах різних виробників та різного призначення. Такі знання можуть стати основою для структурування даних, для систем машинного навчання та для семантичного пошуку. Це викликає потребу у створенні та колективному використанні розподілених баз знань, що базуються на загальноприйнятих стандартах подання інформації. На сьогодні в якості таких стандартів можна використовувати розробки Semantic Web, такі як OWL та RDF, але їх безпосередня підтримка вимагає від користувачів спеціалізованих навичок. Тому виникає необхідність у технологіях, які підтримують колективне накопичення знань та мають достатню виразну потужність для їх інтеграції зі складними інтелектуальними застосунками. Крім того, потрібно орієнтуватися на соціальні засоби подання інформації, які вже сьогодні широко розповсюджені та використовуються багатьма спільнотами користувачів.

Технологія Wiki як основа для колективної побудови бази знань

Важливою рисою багатьох Web-сайтів є створення спільнот, де користувачі зі спільними інтересами є активними учасниками як споживання, так і створення інформації, тобто соціальних мереж. У сучасному розумінні термін «соціальна мережа» використовується для характеристики програмного забезпечення, призначеного для встановлення відношень між окремими особами, а інтелектуалізація Web-технологій визначають тенденцію переходу до семантичних соціальних мереж (S^2N – The Semantic Social Network) [1]. Однією з широко відомих інформаційних технологій, що використовуються для цього, є Wiki, особливістю яких є відкритий підхід до створення контенту [2].

Базові можливості та обмеження Wiki в обробці інформації

Wiki-система – це форма Web-платформи для соціального програмного забезпечення, яка дозволяє спільно створювати, супроводжувати та знаходити цифровий контент. Найбільш вживане нині програмне забезпечення для Wiki-систем – MediaWiki, що базується на PHP і MySQL. MediaWiki використовують проекти Wikipedia, Wikidata, Wikibooks. MediaWiki відрізняється якісним та зручним редактором контенту.

У. Каннінгем, що створив Wiki у 1994 році, визначив цю технологію як *найпростішу онлайнову базу даних*, яка спроможна працювати [3]. Основними характе-

ристиками Wiki і надалі залишаються простота колаборативного використання та редагування сторінок, легке встановлення взаємозв'язків між сторінок всередині і поза Wiki-ресурсом.

Wiki-ресурс – це екземпляр соціального програмного забезпечення, який є сукупністю спільно створених статей, що містять структурований за допомогою Wiki-розмітки текст (призначений для читання та розуміння людьми) і гіперпосилання на інші Wiki-статті або зовнішні IP. Wiki-ресурси є основою для спільного створення та використання знань. Спільними рисами усіх Wiki-ресурсів є наявність Web-інтерфейсу, простий синтаксис для структурування контенту та можливість встановлення гіперпосилань на інші сторінки. Більшість Wiki-систем також забезпечують механізм відкату у випадку випадкових або небажаних змін.

Ефективність використання Wiki як основи для створення інформаційного ресурсу (наприклад, посібника, енциклопедії або довідника) базується на таких особливостях цієї технології, як:

- колаборативність – контент негайно стає доступним для всіх після публікації;
- простота створення документів та інформаційних взаємозв'язків між ними, що полегшує повторне використання інформації;
- тонка деталізація контенту – технологія забезпечує створення окремих сторінок для кожної теми, терміну або слова з довільним рівнем деталізації;
- засоби структурування контенту – за допомогою метаданих (шаблонів, категорій та семантичних властивостей).

Незважаючи на ефективність Wiki як інструменту співпраці, ця технологія має певні недоліки. Інформація, що накопичується у Wiki-ресурсах, є неструктурованою у тому розумінні, що для неї відсутня попередньо визначена модель даних, а її наявні структурні елементи не пристосовані для автоматизованої обробки без додаткових уточнень. Наприклад, природномовна (ПМ) складова контенту має певну лінгвістичну структурованість (поділ на частини мови, члени речення), але у загальному випадку кожна Wiki-сторінка має довільний розмір, складається з довільних частин і не має стандартизованої моделі подання, що викликає складнощі її аналізу.

Така інформація обробляється як неструктуровані дані (НСД) [4], що створює проблеми для менеджменту знань у Wiki-ресурсі. Сторінки не можуть бути автоматично відформатовані відповідно до того, яку інформацію вони містять. Сортування об'єктів, описаних у Wiki за довільними критеріями, потребує використання спеціальних шаблонів.

У традиційних Wiki-ресурсах досить складно імпортувати структуровані дані із зовнішніх баз даних і робити до них запити. Протилежна операція, тобто експорт довільного вибору контенту Wiki до зовнішнього програмного забезпечення, теж пов’язані з певними складнощами та не може бути виконаний автоматично.

Поширений підхід до вирішення цих проблем полягає в тому, щоб семантично структурувати інформацію (з використанням форм та шаблонів) – це робить контент Wiki-ресурсу набагато зрозумілішим для людей і більш доступним для комп'ютерних операцій, таких як пошук на основі структури, інтеграція даних і аналіз даних. Цей підхід називають семантизацією Wiki, і для його реалізації розроблено велику кількість програмного забезпечення, що різниться за своїми цілями, можливостями та вимогами до користувачів. Семантичні Wiki здатні автоматизовано обробляти дані з чітко визначеною семантикою, а це дає змогу істотно розширити функціональність таких систем.

Програмне забезпечення для семантизації Wiki-ресурсів

Для кращого розуміння того, що можна назвати семантизацією Wiki, доцільно порівняти можливості існуючих програмних систем, які більш детально розглянуті в [5, 6].

Приклади семантичного розширення Wiki-технології

AceWiki (http://attempto.ifi.uzh.ch/acewiki) [7] використовує обмежену підмножину англійської мови Attempto Controlled English (ACE). Формальні твердження є основним

вмістом самої Wiki. Таким чином, система намагається інтегрувати онтологію, правила і мову запитів. Редактор дозволяє користувачеві безпосередньо вводити висловлювання або використовувати керовану форму вибору з існуючої онтології. AceWiki концептуалізує кожну Wiki-сторінку як поняття і здійснює виведення OWL для базової онтології.

Kiwi (http://www.kiwi-project.eu/) [8] створює екземпляри даних на основі існуючої онтології, а також надає засоби для створення та редагування онтологій. Kiwi використовує інтерфейс, подібний до Mediawiki, та базується на таких розробках Semantic Web, як RDF і OWL. Формальна структура знань використовується для покращення навігації та надання рекомендацій під час редагування. У системі реалізоване логічне виведення для підтримки користувача у виконанні завдань. Система пропонує і WYSIWYG-редактор, що призначений для користувачів, які не мають досвіду роботи з Wiki-редактором.

KawaWiki [9] надає повну формальну структуру для даних на основі RDF і RDFS. Архітектура Wiki-ресурсу поділяється на три основні шари: 1) схему RDF, що визначає базову онтологію і використовується для перевірки шаблонів RDF; 2) шаблони RDF, які визначають тип сторінок Wiki, які можуть бути створені користувачем; 3) контент Wiki-ресурсу, який користувач отримує для редагування.

OntoWiki [10] призначена для розробки баз знань і спирається на подання даних у форматі RDF. Вона не надає Wiki-інтерфейс для вводу ПМ-тексту для подання понять, але підтримує декілька функцій спільної роботи та дозволяє встановлювати плагіни. OntoWiki реалізує кожну сторінку як ресурс, зберігаючи їх як твердження RDF. Знання в системі презентовані за допомогою «інформаційної карти», збагаченої зручними інтерфейсами для візуалізації й редагування контенту (WYSIWYG редактор для RDF, контроль версій, статистика, підтримка співтовариства тощо). Кожен вузол, поданий як сторінка системи, в інформаційній карті зв’язаний з відповідним цифровим джерелом.

SMW [11] є семантичним розширенням до Mediawiki, що дозволяє користувачам додавати відношення та властивості до Wiki-сторінок. Semantic Mediawiki зберігає семантичні дані в базі даних MySQL у Mediawiki, які також можуть бути експортована як RDF.

Основні семантичні поняття у Semantic MediaWiki – це категорії, що дозволяють користувачам класифікувати Web-сторінки (присутні й у звичайних Wiki-ресурсах), семантичні властивості, які дозволяють встановлювати зв’язки елементів контенту Wiki-сторінок з іншими сторінками Wiki ресурсу та з даними (дата, число, текстовий рядок, географічні координати тощо), визначаючи семантику цих зв’язків, та семантичні запити, в яких можуть використовуватися категорії та семантичні властивості. Результати семантичних запитів вставляються у відповідні Wiki-сторінки та автоматично оновлюються, якщо змінюється контент тих сторінок, з яких вони здобувають інформацію. Це забезпечує цілісність та актуальність створюваного Wiki-ресурсу.

KnowWE є розширенням JSPWiki, що додає до нього семантичну функціональність. Його механізми аналізу та розв'язування проблем також будуються на проекті d3web. Кожна сторінка є поняттям в контрольованій онтології. Семантика включена в Wiki-розмітку і надає три альтернативи: явну розмітку знань, семантичну анотацію та сегментований текст. Текст анотується онтологічним контентом. Крім того, знання для вирішення проблем в тексті можуть бути явними, з використанням, наприклад, правил. Крім того, забезпечується обмін знаннями за допомогою онтологій OWL.

Tiki Wiki CMS Groupware є одним з найбільш багатофункціональних пакетів систем керування контентом (Content Management System – CMS), який забезпечує визначення деякі семантичні відношень між Wiki-сторінками.

Knoodl – це колаборативний редактор Web-онтології. Кожен ресурс є онтологією і має свою власну Web-сторінку, яка містить як структурований контент з онтології, так і неструктурований контент

у вигляді Wiki-тексту. Контент у Knoodl організований у спільноти, які можуть створюватися будь-якими користувачами. Онтології можна імпортувати і експортувати як OWL-файли, з пов'язаним з ними Wiki-текстом або без нього.

Параметри порівняння

Існує багато різних підходів до порівняння засобів семантизації Wiki. З методологічної точки зору при додаванні семантики до Wiki можна виділити два підходи:

- текстові підходи, які збагачують традиційне Wiki-середовище семантичними анотаціями (Semantic MediaWiki);
- логіко-орієнтовані підходи, що використовують семантичні Wiki як засіб для онлайнового редагування онтологій (OntoWiki, Knoodl).

Для того, щоб порівняти різні варіанти семантизації Wiki, в [12] запропоновано два виміри: 1) перспективи користувача – скільки технічних навичок користувач повинен мати для того, щоб використовувати Wiki та додати свій внесок до онтології; та 2) виразність знань – наскільки виразною є результуюча онтологія (рис. 1).

Перспектива користувача (вісь x) розрізняє такі категорії користувачів:

- *Повсякденний користувач* (Everyday user) – Користувач, який знайомий з використанням конкретних програм, без адміністративних навичок, моделювання або програмування;
- *Потужний користувач* (Power user) – користувач, який знайомий з використанням більш широкого кола програм та адмініструванням власного комп'ютера, але без навичок моделювання;
- *Професійний користувач* (Professional user) – досвідчений користувач, який обізнаний у використанні та адмініструванні різноманітного програмного забезпечення, має знання з моделювання та програмування, але не знає технологій Semantic Web (наприклад, OWL і RDF);
- *Онтолог* (Ontologist) – експерт з онтологічного аналізу, який знає, як використовувати технології Semantic Web, зокрема онтології.

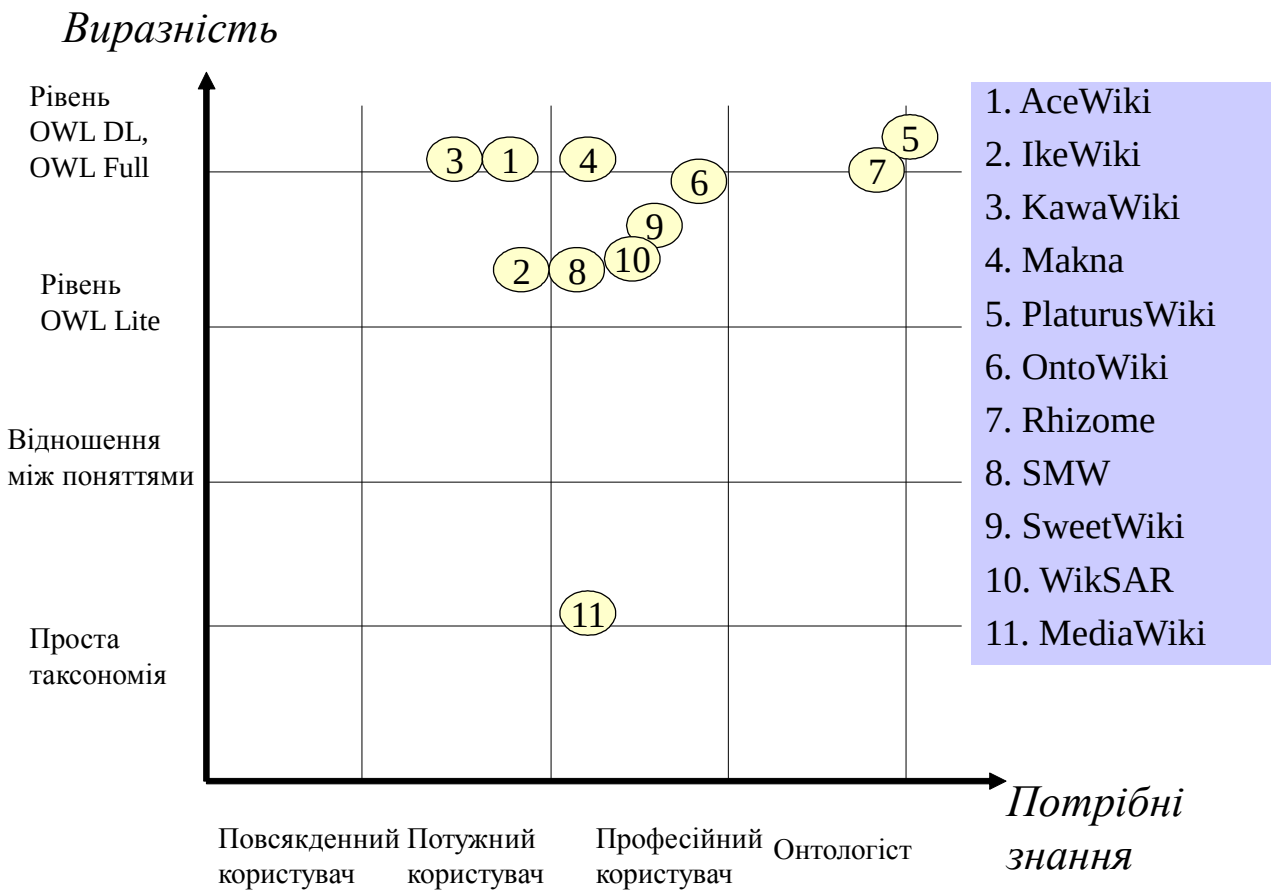


Рис. 1. Порівняння семантичних Wiki

Шкала експресивності (вісь у) виокремлює широкі рівні експресивності та формалізму, які відноситься до того, наскільки виразною є результуюча онтологія:

• *проста таксономія* – таксономічна класифікація Wiki-сторінок без доданої семантики щодо відношень між сторінками та термінами;

• *відношення між поняттями* – поняття в Wiki пов'язані між собою за допомогою семантичних анотацій;

• *рівень OWL Lite* – краща формалізація, ніж прості відношення, з деякими обмеженнями, подібними до функціональності, яку пропонує OWL Lite;

• *рівень OWL DL, OWL Full* – функціональність, подібна до функціональності OWL DL і OWL Full, що підтримує більшу виразність обмежень і відношень з характеристиками властивостей, таких як транзитивність, функціональність, відсутність перетину тощо.

Системи в лівій частині графіка (наприклад, AceWiki і KawaWiki) обмежують дії користувачів і не потребують від них складних навичок, а Wiki у правій частині графіка (KiwiWiki, KnowWE і до деякої міри OntoWiki) мають більшу виразність, але вони потребують від користувачів знання технологій Semantic Web або спеціального синтаксису, а система Knoodl, що знаходиться праворуч, потребує навичок професійного онтолога. Системи, що знаходяться в середній частині графіка, такі як SMW andTiki Wiki CMS Groupware, намагаються збалансувати достатню виразність і знання, необхідні для застосування цієї виразності.

Таке порівняння дозволяє робити припущення щодо знань, що зафіксовані в таких Wiki-ресурсах, та оцінити зусилля користувачів, потрібні для їх створення та підтримки. Більш детально результати порівняння розглянуто в [6].

Це порівняння між семантичними Wiki обґрунтовує вибір MediaWiki і Semantic MediaWiki як базової технології для вирішення питання взаємодії в гетерогенних середовищах, які потребують обміну знаннями та спільного використання:

- з точки зору користувача Semantic MediaWiki базується на тексті і легке в освоєнні і використанні;
- з точки зору виразності знань Semantic MediaWiki являють собою рішення, вдало збалансоване між виразністю та необхідними знаннями.

Semantic MediaWiki використовується у великій кількості як публічних, так і приватних Wiki-ресурсів.

Модель даних у MediaWiki

MediaWiki надає об'єктно-орієнтовану модель зберігання документів середньої деталізації (іменовані фрагменти – це Wiki-сторінки). Модель зберігання подібна в багатьох аспектах на поширені зараз noSQL, хоча попередні MediaWiki зазвичай використовують mysql.

Таким чином, знання створюються і підтримуються в Semantic MediaWiki еволюційно через Web-інтерфейс самою спільнотою користувачів. Загалом, Wiki можна порівняти з системами управління контентом (CMS – content-management systems), що створють та колективно використовують знання, але Wiki надають набагато простіший і більш відкритий інтерфейс для цього технологічного процесу.

Двигун MediaWiki як основа створення Wiki-ресурсів

MediaWiki базується на архітектурі LAMP (PHP на стороні сервера і Javascript на стороні клієнта, з базою даних MySQL). Ядро MediaWiki дозволяє створювати та редагувати сторінки, встановлювати їх категорії та взаємозв'язки, переглядати контент цих сторінок тощо. З точки зору інформаційної структури MediaWiki надає систему шаблонів і можливість категоризації сторінок.

Функціональні шари MediaWiki означають також багаторівневу розробку Wiki:

• *ядро* – базове програмне забезпечення та повний набір API, які надаються для зовнішнього програмного доступу до основних функцій MediaWiki;

• *розширення* – засоби додавання розширених функціональних можливостей до ядра MediaWiki: вдосконалені функції редагування, розширені інтерфейси корис-

тувача та семантичні функції. Зазвичай розширення розробляються мовою PHP, використовуючи механізм перехоплення (hook) для використання подій MediaWiki;

• *шаблони* – дозволяють створювати зразки Wiki з метою зменшення суперечностей у Wiki-ресурсі та гнучкого розвитку схеми;

• *створення контенту* – використання вихідних даних, отриманих від попередніх шарів, та ефективне створення екземпляра MediaWiki, тобто Wiki-ресурсу.

Ця багатошарова структура співпраці в розвитку MediaWiki дозволяє виділити різні групи людей, що приймають участь у створенні Wiki-ресурсу:

- розробники базового програмного забезпечення,
- розробники розширень,
- розробники шаблонів,
- розробники контенту.

З точки зору соціального програмного забезпечення, соціальним об'єктом у MediaWiki є безпосередньо MediaWiki.

З точки зору об'єктно-орієнтованого програмування (ООП), розробка контенту в MediaWiki є створенням екземпляру Wiki-ресурсу.

Основною одиницею Wiki-ресурсу, побудованого на основі MediaWiki, є Wiki-сторінка. У Wiki інформацію можна зберігати на неформальному, напівформальному або формальному рівні, залежно від рівня співпраці спільноти розробників контенту.

MediaWiki надає кілька механізмів для створення структури у Wiki. На цьому рівні дані у Wiki структуровані, напівструктуровані або неструктуровані. За замовчуванням не існує якоїсь обов'язково необхідної структури, але відсутність структурування робить контент дуже складним для навігації. Тому для Wiki-ресурсів, що використовують MediaWiki, зазвичай будується їх власна структура, що відповідає специфіці тієї предметної області, для якої розробляється цей інформаційний ресурс.

Для побудови такої структури в MediaWiki можуть бути використані наступні механізми:

- *Простори імен* (*Namespaces).* Всі сторінки в MediaWiki поділяються на окремі простори імен, що надаються моделлю зберігання. Це дозволяє повторно використовувати базову модель для об'єктів контенту першого класу (наприклад, у е-ВУЕ це сторінки гасел енциклопедії), а також створювати об'єкти, що використовуються при використанні гіпертексту (це можуть бути різноманітні медіа-елементи (двійкові плюс метадані) в просторі імен «Файл», "Ілюстрація" або "Відео", блоки програмування та фрагменти структурованого тексту в просторі імен «Шаблон»);
- *Категорії* (Categories) – це сторінки простору імен Категорії, які можна використовувати для автоматичного групування сторінок. Ці механізми передбачають побудову таксономій у Wiki-ресурсі (наприклад, підкатегорії категорії "Галузь знань" в е-ВУЕ утворюють таксономічну структуру). Категорії Wiki відповідають визначенню класу в об'єктно-орієнтованій парадигмі, а сторінки Wiki-ресурсу, які належать до певних категорій, є екземплярами цих класів;
- *Шаблони* (*Templates).* MediaWiki надає простір імен, присвячений транзакції, який називається Шаблон. Сторінки у цьому просторі назв призначені для включення до інших статей повторюваних елементів. Цей механізм використовується також для уникнення суперечностей у Wiki та для більш уніфікованого подання інформації. Це поширений спосіб структурування даних. Наприклад, в е-ВУЕ створено шаблони для опису типових інформаційних об'єктів.

Використання шаблонів структурування контенту Wiki-ресурсу

Модель на основі шаблонів передбачає гнучку розробку схеми Wiki-ресурсу. Кожен шаблон визначає клас з вільно обраними атрибутами (еквівалент "тип сутності"), екземпляри яких можуть бути вільно вбудовані в інші об'єкти. Екземпляри шаблонів можуть бути пов'язані ієрархічно. Наприклад, у е-ВУЕ за допомогою набору ієрархічно організованої множини шаблонів подаються знання щодо типових інформаційних об'єктів (ІО).

Механізми шаблонізації контенту використовуються у Wiki для того, щоб дати авторам контенту можливість заповнити схему (модель) конкретними значеннями. Сторінки з такими шаблонами можуть використовуватися для опису подібних елементів.

Вивчення сучасних механізмів Wiki-шаблонів дозволяє зрозуміти, як саме вони можуть бути використані для створення семантично багатих Web-сторінок. При цьому можна виділити дві альтернативні моделі шаблонів, які можна знайти в сучасних Wiki-движках:

− *функціональні* шаблони (functional templating), коли шаблони викликаються за ім'ям і переданими параметрами;

− шаблони *створення* (creational templating), в яких шаблони просто копіюються як новий контент на початок історії перегляду сторінки.

На відміну від традиційних CMS, де шаблони розглядаються як спеціальні ресурси, у Wiki-технології шаблони та екземпляри розглядаються на одному рівні: це дозволяє користувачам керувати обома видами інформації подібним чином.

В обробці шаблонів беруть участь два види Wiki-сторінок:

- *сторінка шаблону* (наприклад, *Шаблон:Річка*): основна сторінка, інформацію з якої використовувати ті сторінки, на яких цей шаблон викликається;
- *сторінка екземпляру шаблону* (наприклад, *Дніпро*): сторінка, контент якої побудовано з використанням виклику шаблону з певними конкретними значеннями його параметрів.

У функціональних шаблонах слід відмітити кілька важливих характеристик:

- міцний і постійний зв'язок між шаблоном та його екземплярами (зміни, що вносяться на сторінку шаблону, відразу ж відображаються на усіх сторінках, де цей шаблон викликається);
- різний вихідний код шаблонів та їх екземплярів;
- екземпляри шаблонів є нелінійною розміткою щодо її форми інтерпретації.

Важливо, що за допомогою шаблонів дані, що містяться на Wiki-сторінках, можуть бути структуровані безпосередньо – шляхом їх зв’язування із семантичними властивостями та з категоріями. Незважаючи на це, такі дані, вбудовані в екземпляри шаблонів, складно вилучати і повторно використовувати, хоча різні дослідники пропонують для виконання цієї задачі досить ефективні евристичні методи. Наприклад, один з таких методів використовується у проекту Dbpedia [13], що спрямований на здобуття структурованої інформації з Вікіпедії та забезпечення доступності цієї інформації в Web-середовищі.

У Вікіпедії, яка не є семантизованим ресурсом, нині теж досить часто застосовуються шаблони. Найчастіше такі шаблони призначені лише для макетування інформації, однак відносна частка шаблонів, що забезпечують структурування контенту через елементи Wiki-розмітки, що відмінні від семантичних властивостей, постійно збільшується (приблизно від чверті до третини сторінок Вікіпедії нині містять структуровану інформацію, придатну для машинної інтерпретації). Наприклад, широко застосовується особливий тип шаблонів – інфорбокси (infoboxes), що призначені для створення послідовно форматованих блоків для певного контенту в статтях, які описують екземпляри певного специфічного типу (наприклад, типові ІО). Щоб розкрити семантику, закодовану в таких шаблонах, використовують різні методи здобуття знань. Застосування таких методів може бути корисним і для семантизованих Wiki-ресурсів, тому що вони забезпечують отримання тих знань, що неявно містяться в контенті, але не формалізовані повністю для автоматичного видобуття. Наприклад, в [14] запропоновано такий алгоритм здобуття інформації, що працює в п'ять етапів.

1. Обрати усі сторінки Wiki-ресурсу, які містять шаблони. Таку множину сторінок Wiki-ресурсу можна отримати за допомогою запиту SQL, який шукає входження роздільника шаблону "{{" в тексті бази даних MediaWiki. Запит SQL дозволяє також вибрати тільки сторінки певних категорій Wiki-ресурсу або сторінки, що містять певні шаблони.

2. Здобути з обраних сторінок важливі шаблони. Усі шаблони на сторінці Wiki-ресурсу можна отримати за допомогою рекурсивного регулярного виразу. Через те, що шаблони використовуються у Wiki-ресурсі для різних цілей, потрібно обрати тільки ті шаблони, що містять структуровану інформацію. Це можна зробити на основі такої евристики: ігнорувати шаблони з одним або двома атрибутами (такі шаблони зазвичай діють як ярлики для попередньо визначених розділів), а також ігнорувати шаблони, кількість використання яких не перевищує певний поріг (такі шаблони можуть бути помилковими або неважливими). Алгоритм здобуття може бути додатково сконфігуровано для ігнорування певних явно визначених шаблонів або груп шаблонів, наприклад, якщо ці шаблони пов'язані з іншими аспектами структурування інформації, такими як умови створення та використання контенту.

3. Виконується аналіз структури кожного обраного шаблону і створюються відповідні трійки "суб'єкт-предикат-об'єкт". URL-адреса, отримана з назви Wiki-сторінки, на якій знайдено шаблон, використовується як суб'єкт для шаблонів, які використовуються не більше одного разу на сторінці. Для шаблонів, що зустрічаються на сторінці більше за один раз, потрібно генерувати новий ідентифікатор, який використовується як суб'єкт. Кожен атрибут шаблону відповідає предикату трійки, а відповідне значення атрибута перетворюється на його об'єкт. Шаблони MediaWiki можуть бути вкладеними, тобто значення атрибута в шаблоні може знову бути шаблоном. У такому випадку генерується порожній вузол, що пов'язує значення атрибута з новостворюваним екземпляром для вкладеного шаблону.

4. Обробка значень об'єктів для створення відповідних URI-посилань або літеральних значень. Для гіперпосилань MediaWiki створюються відповідні URI-посилання, які забезпечують перехід на пов'язану Wiki-сторінку. Типізовані літерали генеруються для рядків і числових значень. Загальновживані одиниці виміру (наприклад, кг для кілограмів, с для секунд) виявляються і кодуються як спеціальні типи даних, однак перетворення між різними масштабами (наприклад, між см, м, км) не виконується автоматично. Крім того, розпізнаються списки, що розділяються комами, і, залежно від параметрів конфігурації, перетворюються на списки RDF або на окремі висловлювання.

5. Визначення належності Wiki-сторінки до класу. Сторінки Wiki-ресурсу організовані за категоріями. У деяких випадках ці категорії можуть бути інтерпретовані як класи, що включають екземпляри, описані сторінками Wiki-ресурсу з відповідній категорії. Крім того, назва шаблону може бути індикатором належності до класу. Самі категорії є сторінками Wiki-ресурсу і також можуть бути організовані у надкатегорії (такі відношення можуть відображатися через зв'язок «бути пов'язаними з»).

На основі використання цього методу здобуття знань з Wiki-ресурсів, що структуровано за допомогою шаблонів, його авторами розроблено набір рекомендацій для побудови Wiki-шаблонів. Дотримання цих рекомендацій не тільки корисно для семантичного здобуття інформації з Wiki-сторінок, але зазвичай також покращує відповідний шаблон у цілому, тобто він стає більш зручним для використання авторами Wiki-сторінок. Будуючи нові шаблони та модифікуючи існуючі, доцільно дотримуватися наступних правил:

- не визначати атрибути в шаблонах, які кодують інформацію для макетування сторінки – це правило відповідає відомому принципу відокремлення контенту від його подання, тобто HTML-розмітка повинна використовуватися в значеннях атрибутів тільки у разі необхідності;
- використовувати лише один шаблон для певного інформаційного об'єкта замість того, щоб використовувати окремий шаблон для кожного атрибута (це потребує створення більш складних шаблонів, але спрощує процес здобуття знань щодо структури цих об'єктів);
- кожен атрибут шаблону повинен мати точно одне значення в шаблоні статті (хоча таке значення може бути спи-

ском значень), але не слід змішувати декілька операторів (з точки зору RDF) в межах одного значення атрибута (тобто потрібно більш чітко відрізняти різні атрибути шаблона, виокремлюючи їх можливі значення);

- не бажано використовувати різні шаблони для однакової цілі (таке дублювання структури різними шаблонами не тільки ускладнює здобуття знань й потребує додаткових операцій з їх об'єднання, але й спричиняє накопичення застарілих шаблонів із некоректно поданою інформацією);
- не слід використовувати різні імена атрибутів для одного і того ж типу контенту (наприклад, в різних шаблонах називати подібний атрибут "Назва міста", "Назва селища" та "Позначення міста") і не доцільно використовувати одне ім'я атрибута для різних типів вмісту (наприклад, в шаблонах "Персоналія" та "Місто" використовувати атрибут "Ім'я");
- намагатися використовувати стандартні подання для елементів, щоб вони могли бути виявлені алгоритмом здобуття.

Крім того, для більш зручного здобуття знань та коректного введення даних у Wiki-ресурсах доцільно мати наступні можливості:

- зафіксувати тип даних для значення атрибута шаблону. У Semantic MediaWiki це можна робити за допомогою типів значень семантичних властивостей, але тільки для базових типів даних. Наприклад, можна вказати у визначенні шаблону, що значення атрибуту "Рік народження" має бути числом, а "Місце народження" – посиланням на Wiki-сторінку. Але необхідно враховувати, що іноді виникає необхідність використовувати природномовні описи для значень атрибутів, наприклад, значення року народження може бути "у середині 3 століття" або "приблизно між 1100 та 1105 роками", що призводить до застосування менш жорстких обмежень на дані, наприклад, усі дані описуються як текстовий рядок;
- використовувати специфічні для певного Wiki-ресурсу типи даних, наприклад, забезпечити засоби введення тегів природної мови (якщо певне значення атрибута може бути подано різними мовами, наприклад, прізвище особи може бути подано українською та рідною мовою особи) та одиниць виміру для елементів, які можуть оцінюватися у різних одиницях (наприклад, у метрах та футах).

Для атрибуту, який визначено у шаблоні, MediaWiki дозволяє знайти шаблони, де цей атрибут вже використовується, та показувати інші характеристики атрибуту, щоб дати розробнику шаблону можливість перевірити, чи мають ці атрибути однакове семантичне значення. Це спрощує розробку шаблонів та дозволяє уніфікувати імена параметрів, не викликаючи суперечностей. Наприклад, в e-BYE атрибут "Площа" має однакові одиниці виміру та тип значення для шаблонів "Країна" та "Озеро", тому доцільно використовувати в них однакову назву. Але використання атрибуту "Ім'я" в шаблонах "Персоналія" та "Літературний твір" недоцільно, якщо в першому випадку цей параметр має значення типу "текстовий рядок" та характеризує особисте ім'я особи, а в другому – це посилання на прізвище автора, що має тип "Wiki-посилання", і тому краще замінити його на параметр "Автор".

Модель даних Semantic MediaWiki

Semantic MediaWiki – це семантично розширений Wiki-двигун, який дозволяє користувачам анотувати контент Wiki явно визначеною інформацією, придатною для автоматичної обробки (машиночитаною). Він підтримує додавання структурованої та семантично анотованої інформації у Wiki з використанням відповідного синтаксису.

Semantic MediaWiki (SMW) є надбудовою над інструментальним засобом побудови Wiki-сайту MediaWiki. Переваги SMW – це обробка інформації на семантичному рівні, наявність засобів групового керування знаннями, відносно висока виразна потужність, надійна реалізація і зручний інтерфейс користувачів, наявність документації та спільнот користувачів. SMW дозволяє інтегрувати інформацію з різних Wiki-сторінок, здійснюючи пошук

на рівні знань, та генерувати за Wiki-сторінками онтологічні структури, які можуть використовувати інші системи.

Використовуючи семантичні елементи, SMW вирішує такі основні проблеми сучасних Wiki:

- послідовність контенту (*Content consistency)*: інформація, що зустрічається на різних Wiki-сторінках, має бути несуперечною.
- доступність знань (*Knowledge Access*): пошук і порівняння інформації з різних сторінок у Wiki-ресурсі великого обсягу потребує відповідних засобів.
- повторне використання знань (*Knowledge reuse*): на відміну від жорсткого контенту на основі тексту у традиційних Wiki, який може використовуватися лише для читання сторінок у браузері або подібній програмі, семантизація є основою для доступу до Wiki-контенту з інших застосунків та експорт знань у поширені формати їх подання.

Крім категорій, в SMW для структурування інформації використовуються такі механізми, як *семантичні властивості*. Вони дозволяють семантично пов'язувати Wiki-сторінки як між собою, так і з різними даними. Кожна семантична властивість має тип, назву і значення, а також власну Wiki-сторінку в спеціальному просторі імен, яка дозволяє визначати її місце в ієрархії властивостей та документувати те, як цю властивість необхідно використовувати. Кожна семантична властивість має власну Wiki-сторінку. На цій сторінці можна явно визначити тип значень, що приймає ця властивість, використовуючи конструкцію [[Has type::xxx]]. Has type розпізнається SMW як спеціальна властивість, тобто значення такої властивості є посиланням на іншу Wiki-сторінку. В Semantic MediaWiki підтримуються наступні типи властивостей: String, Number, Annotation UR1, Boolean, Email, Text, Geographic Coordinates тощо.

Динамічно генерований контент може створюватися за допомогою вбудовування запитів у Wiki-сторінки. Імпорт зовнішніх даних з існуючих онтологій типу FOAF, SIOC або Doublin Core може здійснюватися шляхом зіставлення Wiki-анотацій з елементами цих словників.

Щоб зробити можливим зовнішнє повторне використання інформації з Wiki-ресурсу, що базується на Semantic MediaWiki, можна отримувати формальний опис для однієї або кількох статей через Web-інтерфейс у форматі OWL/RDF. Оскільки SMW чітко дотримується стандарту OWL DL, експортовану інформацію можна повторно використовувати в різних інструментах.

Синтаксис семантичних анотацій використовує мову сценаріїв для опису контенту сторінок Wiki, який розширюється в SMW за допомогою наступних трьох основних наборів семантичних анотацій.

Визначення класів і властивостей: SMW повторно використовує простір імен "Category" ("Категорія") для визначення класів. Наприклад, сторінка Wiki з назвою "Category: Images" призначена для подання класу всіх зображень, а також новий простір імен під назвою "Властивість" для визначення властивостей понять. Завдяки такому підходу, SMW підтримує як бінарні, так і n-арні властивості.

Аксіоми: SMW дозволяє декларувати відношення "клас-підклас" і "властивість-субвластивість". Наприклад, можна декларувати, що категорія "Зображення" є підкатегорією "Медіа", додавши на сторінку цієї категорії анотацію "[[Category: Медіа]]". Це дозволяє утворювати таксономію категорій та властивостей, яка є дуже корисною як для навігації у Wiki-ресурсі, так і для коректного створення нових категорій та властивостей. SMW також підтримує затвердження еквівалентності між двома Wiki-сторінками або між двома класами за допомогою механізму переспрямування (redirection) у MediaWiki. Для цього використовується конструкція «REDIRECT [[x]]. Наприклад, в e-BYE за допомогою цього механізму організовано відсильні статті, коли існує кілька загальновживаних формулювань того самого терміну.

Твердження про екземпляри: SMW дозволяє оголошувати сторінку екземп-

ляром класу або суб'єктом RDF-трійки. Наприклад, якщо сторінка Wiki «Зображення Хрещатику» містить анотацію «[[Категорія: Зображення]] та [[Type :: Фотографія]]», це означає, що Зображення Хрещатику є цифровим ресурсом і має тип «Фотографія». Також можна створити екземпляр n-арної властивості. Наприклад, на Wiki-сторінці "Зображення Хрещатику" анотація "[[Загальні ключові слова :: Зображення Хрещатику; пейзаж]]" означатиме, що Зображення Хрещатику відноситься до предметної категорії пейзажей.

З точки зору функціональності механізм семантичних анотацій відображає три основні аспекти функціональності:

• функціональність *гіпертекстових посилань* – успадкована від MediaWiki для встановлення зв'язків між Wiki-сторінками;

• функціональність *подання знань* у семантичну мережу;

• функціональність бази даних, що забезпечує подання n-арних кортежів властивостей в SQL-подібному стилі.

Шаблони в Semantic MediaWiki – це Wiki-сторінки в спеціальному просторі імен. Використання шаблонів корисне для структурованого введення інформації користувачами. Агрегація властивостей в шаблонах та агрегація шаблонів подані у семантичних формах.

Приклад форми:

Property: CollectionName | hasType: String Property: BestQualityURI | hasType: URL Property: Type| hasType: String

Приклад шаблону:

{{Metadata | CollectionName = | BestQualityURI = | Type = }}

Form: {{#forminput:Form-MetadataEntry

{{{for Metadata

{| class="formtable" | Collection Name: | {{{field | CollectionName}}}

...

{{{end template}}}

}}

Для пошуку Semantic MediaWiki використовує вбудовані запити.

Семантичні властивості та вбудовані запити разом із семантичними формами та шаблонами є потужним інструментом, який може підтримувати моделювання для різних потреб проектів.

Структура семантизованого Wiki-ресурсу як основа для побудови його онтологічної моделі

Використання семантичних Wiki-технологій для створення розподілених інформаційних ресурсів не тільки дозволяє досить легко додавати структурування до неструктурованих даних (НСД), але й є джерелом фонових знань для аналізу довільних природномовних текстів відповідної предметної області. Створення е-ВУЕ як семантизованого Wiki-ресурсу дозволяє вдосконалити процес генерації таких знань. Використання онтологічного аналізу є основою для переходу від неструктурованого контенту [4] до розподіленої бази знань, придатної для повторного використання.

З точки зору онтологічного аналізу, кожна Wiki-сторінка являє собою онтологічний елемент, тобто елемент одного з RDF-класів – Thing, Class, ObjectProperty, DatatypeProperty, AnnotationProperty. Крім того, кожна стаття має власний URI, який дозволяє уникнути плутанини між поняттями і HTML- сторінками. Зазвичай, статті є екземплярами класів онтології OWL, категорії – класами, а відношення – об'єктними властивостями онтології.

Виходячи з цього, для будь-якої сторінки SMW за запитом може генерувати відповідний OWL/RDF-файл. Найпростіший спосіб отримати цей RDF – використати посилання "Переглянути як RDF" ("View as RDF"), що знаходиться в нижній частині кожної анотованої сторінки. Ця сторінка може виступати як кінцева точка (endpoint) для зовнішніх сервісів (зовнішньої точки доступу), які хочуть отримати доступ до семантичних даних SMW. На жаль, ця функція реалізована дуже невдало та підтримує надто мало опцій.

Оскільки SMW сумісна з моделлю знань OWL DL, то існує можливість використання в Wiki існуючих онтологій. Це можливо здійснити двома шляхами: імпорт онтології дозволяє створювати і модифікувати сторінки в Wiki для подання відношень, заданих в деякому існуючому OWL DL-документі; а повторне використання словника дозволяє користувачам відображати (задавати відповідності) Wiki-сторінки на елементи існуючих онтологій.

Функція імпорту онтології для читання RDF-документів використовує інструментарій RAP toolkit. Він витягує RDF-твердження, які можуть бути подані у Wiki. Найменування статей імпортованих елементів витягуються з їх міток (labels), або, в разі відсутності мітки, з ідентифікатора розділу їх URI. Основною метою імпорту є ініціалізація (первинне автоматичне завантаження) основи-шаблону для заповнення Wiki. Крім того, імпорт онтології вставляє спеціальні анотації, які генерують еквівалентні твердження в експорт OWL (тобто. owl:sameAs, owl:equivalentClass, або owl:equivalentProperty). Імпорт онтологій дозволений тільки для адміністраторів сайту, оскільки це може бути використано для спаму.

Імпорт словника дозволяє користувачам ідентифікувати елементи Wiki, вказавши зв'язок з елементами існуючих онтологій. Наприклад, Category:Person може безпосередньо експортуватися в клас foaf:Person словника Friend-Of-A-Friend. Wiki-користувачі можуть вирішувати, які сторінки Wiki повинні мати зовнішню семантику, проте набір наявних зовнішніх елементів управляється тільки адміністраторами. Вводячи в словник Wiki деякий новий елемент, вони повинні упевнитися в тому, що повторне використання словника співвідноситься з типами обмежень OWL DL. Наприклад, зовнішні класи, такі, як foaf:Person, не можуть бути імпортовані у відношення.

Експорт в OWL/RDF є засобом забезпечення зовнішнього повторного використання даних з Wiki, але тільки практичне застосування цієї функції може показати якість згенерованого RDF. З цією метою для видачі RDF розробники системи використовували ряд інструментів Semantic Web. SMW добре співпрацювало з найбільш відтестованими застосуваннями, такими, як FOAF Explorer, Tabulator RDF browser, або розширенням браузера Piggy Bank RDF.

Крім того, SMW надає сервіс для підтримки SPARQL запитів. Система базується на автономному (stand-alone) RDF сервері Joseki, який синхронізований з семантичним контентом Wiki. Синхронізація полягає у генерації RSS-фіду зі звітом про останні зміни, що відбулися у Wiki, для того, щоб швидко перезавантажити змінені статті. Таким чином, SPARQL-точка доступу (endpoint) демонструє можливість дзеркально відобразити RDF-контент Wiki за допомогою невеликих покрокових оновлень, і пропонує точку доступу для семантичних технологій, які повторно використовують ці дані.

Формальна модель Wiki-онтології

Wiki-онтологія – це онтологія, що відображає знання семантично розміченого Wiki-ресурсу (набору Wiki сторінок, що містять семантичну розмітку) [15]. Вона містить тільки ті знання, які можна безпосередньо здобути із семантичної розмітки. Тому в цій онтології відсутні, приміром, такі характеристики класів та властивостей, як еквівалентність, відсутність перетину тощо. Така онтологія може бути згенерована в результаті аналізу Wiki-ресурсу або, навпаки, створена до початку розробки самого ресурсу та використовуватися як основа для його семантичної розмітки, тобто її поняття та їх атрибути використовуються для позначення категорій та семантичних властивостей Wiki-ресурсу. На практиці зазвичай обидва варіанти використовуються ітеративно – спочатку генерується онтологія предметної області, потім вона використовується у розмітці, а в процесі наповнення контентом Wiki-ресурсу вносяться вдосконалення і в саму онтологію.

Для опису формальної моделі Wiki-онтології пропонується використовувати наступну формальну модель онтології $O = \langle X, R, F, T \rangle$ [16], що складається з наступних елементів:

- $X = X_{cl} \cup X_{ind}$ – множина концептів онтології, де
  - X_{cl} – множина класів,
  - X_{ind} – множина екземплярів класів, таких, що $\forall a \in X_{ind} \exists A \in X_{cl}, a \in A$;
- $R = r_{ier\_cl} \cup \{r_i\} \cup r_{ier\_prop} \cup \cup \{p_j\} \cup p_{ier\_prop}$ – множина відношень між елементами онтології, де
  - r_{ier_cl} – ієрархічні відношення між класами онтології – це структури часткового впорядкування з верхнім елементом *Thing*, що можуть встановлюватися між класами онтології і характеризується такими властивостями, як антисиметричність і транзитивність, $r_{ier\_cl} : X_{cl} \to X_{cl}$;
  - $\{r_i\}$ – множина об'єктних властивостей, що встановлюють відношення між екземплярами класів: $r_i(a, a \in X_{ind}) = b, b \in X_{ind}$, $r_i : X_{ind} \to X_{ind}$;
  - r_{ier_prop} – ієрархічні відношення між об'єктними властивостями класів онтології – це структури часткового впорядкування з верхнім елементом *topObjectProperty*, що можуть встановлюватися між класами онтології і властивостями класів і характеризується такими властивостями, як антисиметричність і транзитивність, $r_{ier\_prop} : \{r_i\} \to \{r_i\}$;
  - $\{p_j\}$ – множина властивостей даних, що встановлюють відношення між екземплярами класів і значеннями з T: $p_i(a, a \in X_{ind}) = t, t \in T$, $p_i : X_{ind} \to T$;
  - p_{ier_prop} – ієрархічні відношення між властивостями даних екземплярів класів онтології – це структури часткового впорядкування з верхнім елементом *topDataProperty*, що можуть встановлюватися між властивостями даних екземплярів класів онтології і характеризується такими властивостями, як антисиметричність і транзитивність, $p_{ier\_prop} : \{p_i\} \to \{p_i\}$;
- $F = \{F_{cl} \cup F_{prop}\}$ – множина характеристик, що можуть використовуватися для логічного виведення над онтологією:
  - F_{cl} – множина характеристик класів онтології, що можуть застосовуватися для логічного виводу: еквівалентність, відсутність перетину тощо;
  - F_{prop} – множина характеристик об'єктних властивостей екземплярів класів онтології, що можуть застосовуватися для логічного виводу: транзитивність, симетричність, антисиметричність, рефлексивність, антирефлексивність тощо);
- T – множина типів даних (наприклад, рядок, ціле), значення з яких можуть приймати властивості даних класів онтології;
- M – множина нелогічних правил ПрО (наприклад, рядок, ціле).

У Wiki-онтології O_{wiki} множина концептів будується як поєднання таких елементів Wiki, як сторінки та категорії $X = X_{wiki\_categor} \cup X_{wiki\_page}$, пов'язаних різними видами відношень з $R = \{r_{ier\_cl}\} \cup \{r_{link}\} \cup \{r_{sem\_prop}\}$: множина класів – це множина категорій Wiki $X_{wiki\_categor}$, між якими існують ієрархічні відношення r_{ier_cl}; множина екземплярів – множина Wiki-сторінок X_{wiki_page}, між якими існують посилання r_{link} та семантичні відношення $r_{sem\_prop_i}, i = \overline{0,m}$; множина типів даних доповнюється специфічним класом – "Wiki-сторінка". Ця модель може бути вдосконалена з урахуванням таких елементів Wiki, як шаблони, форми, спеціальні сторінки тощо.

У формальній моделі Wiki-онтології O_{wiki} не семантизованого Wiki-ресурсу X ці множини мають наступний склад:

- X – це множини сторінок Wiki-ресурсу:

$$X_{wiki} = P_{user} \cup P_{categ} \cup P_{template} \cup P_{spec},$$

де

- P_{user} – множина сторінок, створених користувачами,

- P_{categ} – множина сторінок, що описують категорії,
- $P_{template}$– множина сторінок, що описують шаблони,
- P_{spec} – множина інших спеціальних сторінок;

– $R = r_{ier\_categor} \cup R_{wiki}$

- $r_{ier\_categor}$ – ієрархічні відношення, що можуть встановлюватися між категоріями Wiki-ресурсу і характеризується такими властивостями, як антисиметричність і транзитивність,

$$r_{ier\_categor}: P_{categ} \to P_{categ};$$

- $R_{wiki} = \{"link","is\_a","use"\}$ – множина з трьох елементів, де "link" – відношення, що описує посилання однієї довільної Wiki-сторінки цього ресурсу на іншу Wiki-сторінку цього ресурсу (хоча в Wiki-ресурсах передбачені і посилання на інші види сторінок, у рамках даної моделі вони не враховуються), $"is\_a": X \to P_{categ}$ – відношення включення довільної сторінки до певної категорії, а $"use": X \to P_{template}$ – відношення, що формалізує включення Wiki-шаблону до довільної сторінки ресурсу;

– T – множина типів даних, що підтримуються в MediaWiki (наприклад, рядок, ціле, число, дата, координата), значення з яких можуть приймати властивості даних класів онтології;

– інші множини такої Wiki-онтології є порожніми.

Формальна модель Wiki-онтології O_{s_wiki} для семантично збагачених Wiki-ресурсів є більш складною і включає ряд елементів, зв'язаних із семантичними властивостями [16]:

– X – це множини сторінок Wiki-ресурсу:

$$X_{sem\_wiki} = X_{wiki} \cup P_{s\_prop},$$

яка доповнюється множиною P_{sem_prop} – сторінками семантичних властивостей, деякі з який є семантично визначеними посиланнями на інші Wiki-сторінки:

$$P_{sem\_prop\_page} \subseteq P_{sem\_prop},$$

а інші зв'язують сторінки зі значеннями різних типів даних (ці типи даних визначаються на сторінках семантичних властивостей);

– $R = r_{ier\_cl} \cup R_{semant\_wiki}$, де

- $r_{ier\_cl} = r_{ier\_categor} \cup r_{ier\_property}$ розширюється порівняно із онтологією несемантизованого Wiki-ресурсу введенням окремого ієрархічного відношення $r_{ier\_property}$ для задання ієрархії семантичних властивостей;
- $R_{semant\_wiki} = R_{wiki} \cup \{"has\_property", "property\_value"\}$ – множина відношень, що доповнюється відношеннями, що пов'язані із семантичними властивостями Wiki-сторінок:

$$"has\_property": X \to P_{sem\_prop}$$

– відношення, що вказує на те, що довільна Wiki-сторінка має певну семантичну властивість, та

$$"property\_value": P_{sem\_prop} \to X \cup T$$

– відношення, що пов'язує семантичну властивість сторінки з її значенням, яке може бути як посиланням на іншу сторінку, так і літералом;

– T – множина типів даних, що підтримуються в MediaWiki;

– інші множини такої Wiki-онтології є порожніми.

Вибір саме такої моделі онтології для даної задачі обумовлюється наступними причинами. По-перше, така модель має достатню виразність для розв'язання широкого класу інтелектуальних задач. По-друге, вона відповідає інтуїтивному уявленню про онтології, яке міститься в користувальницькому інтерфейсі редактора онтологій Protégé і тому легко поєднується з візуалізаціями елементів онтології в цьому програмному продукті. По-третє, ця модель досить легко інтегрується з різними інтелектуальними застосуваннями,

які підтримують семантичну обробку інформації (приміром, з семантичними Wiki-ресурсами, лексичними онтологіями).

Онтологічна модель бази знань е-ВУЕ

Розглянемо проаналізовані вище засоби побудови та вдосконалення структури бази знань Wiki-ресурсу на прикладі науково-інформаційного наповнення портальної версії Великої української енциклопедії – е-ВУЕ (http://vue.gov.ua), що використовує вільне програмне забезпечення MediaWiki версії 1.29.1. та його семантичне розширення Semantic MediaWiki версії 2.5.5 – інноваційний проект зі створення національної енциклопедії на основі сучасних засобів подання знань.

Принциповими відмінностями е-ВУЕ від відомих онлайнових довідників та енциклопедій (приміром, від Вікіпедії) є: 1) орієнтація на авторські статті, тобто на оригінальний та якісний контент, підготовлений експертами відповідної області; 2) наявність складної системи семантичних відношень між гаслами, що базуються на наборах ієрархій категорій та семантичних властивостей; 3) рецензованість гасел, що забезпечує більший рівень об'єктивності та довіри до поданої інформації.

Через високу складність інформаційного наповнення е-ВУЕ виникає необхідність застосування засобів онтологічного аналізу для моделювання структури, взаємозв'язків та властивостей об'єктів, що складають контент цієї енциклопедії. Ці засоби дозволяють встановлювати співставлення між Wiki-онтологією (тобто тією онтологією, елементи якої лежать в основі семантичної розмітки Wiki-ресурсу) та довільною онтологією ПрО. Онтологічна модель знань е-ВУЕ дозволяє перетворити її на розподілену базу знань, що є джерелом корисної та перевіреної інформації як для людей, так і для інтелектуальних програмних засобів, необхідно створити цього енциклопедичного видання. Така модель формалізує відношення між її основними об'єктами, їх типами та властивостями.

Категорії е-ВУЕ

Кожне гасло е-ВУЕ може бути віднесено до довільної кількості категорій. Засоби Wiki-середовища дозволяють явно вказувати ієрархічні ("є підкатегорією") зв'язки між такими категоріями. Жодних зовнішніх обмежень на такі зв'язки в Semantic MediaWiki немає. Такі категорії можуть відображати різні аспекти, за якими можна класифікувати гасла енциклопедії. Деякі з них відображають специфіку предметної області енциклопедистики, інші можуть стосуватися, приміром, умов публікації та використання матеріалу.

Можна виділити кілька незалежних ієрархій категорій е-ВУЕ:

- напрямки знань та їх підкласи (приклад ієрархії підкатегорій: "Технічні науки" – "Радіотехніка та телекомунікації" – "Оптоелектронні системи");
- тип інформаційного об'єкта – "Персоналії", "Цивілізація" та "Природа", а також підкатегорії цих категорій;
- тип опублікування – категорії "е-ВУЕ" (всі сторінки гасел енциклопедії) та її підкатегорія "ВУЕ" (інформація, що опублікована у паперовій версії ВУЕ);
- мультимедійні матеріали;
- типи учасників побудови контенту – категорії "Автори ВУЕ", "Модератори".

З точки зору тематики найвищим рівнем класифікації в ієрархії категорій є поділ гасел на три категорії, що не перетинаються: "Персоналії", "Цивілізація" та "Природа". Усі повністю завершені гасла на порталі е-ВУЕ віднесено до прихованої категорії "е-ВУЕ", яка використовується в семантичних запитах і дозволяє не брати до розгляду та аналізу технічні й допоміжні сторінки, – це сторінки е-ВУЕ, на які можна посилатися із зовнішніх джерел.

Шаблони е-ВУЕ

Шаблони Semantic MediaWiki дозволяють автоматизувати введення відповідної інформації на сторінках та забезпечують уніфікацію імен категорій та семантичних властивостей. Крім того, вони дозволяють інтегрувати контент різних сторінок (з використанням убудованих запитів) та запобігати суперечностей у поданні інформації.

В процесі розробки енциклопедії виділено типові інформаційні об’єкти (ІО) – Wiki-сторінки з подібним набором розділів та близьким набором семантичних властивостей. Для таких типових ІО створено Wiki-шаблони, що дозволяють уніфіковано відображати відомості на екрані та забезпечують коректне введення значень семантичних властивостей.

На сьогодні в е-ВУЕ використовуються наступні шаблони базових типових ІО (рис. 2):

Битва	Відзнаки
Війна	Група осіб
Історична подія	Книжкове видання
Країна	Материк
Мінерал	Місто
Модератор	Море
Музичний гурт	Озеро
Океан	Олімпійські_ігри
Організація	Періодичне видання
Персоналія	Порода
Рельєф	Річка
СМТ	Таксон

Семантичні властивості е-ВУЕ

В е-ВУЕ використовуються семантичні властивості різних типів, які дозволяють як пов’язувати Wiki-сторінки на рівні знань, так і фіксувати значення їх атрибутів. Для кожного з типових ІО визначено набір таких властивостей та задано за допомогою шаблонів форму їх відображення. Наприклад, шаблон “Озеро” має такі семантичні властивості, як “Найбільша глибина” (число, єдине значення), “Країни” (посилання, можливо кілька значень), “Тип живлення” (текст) та “Площа” (число).

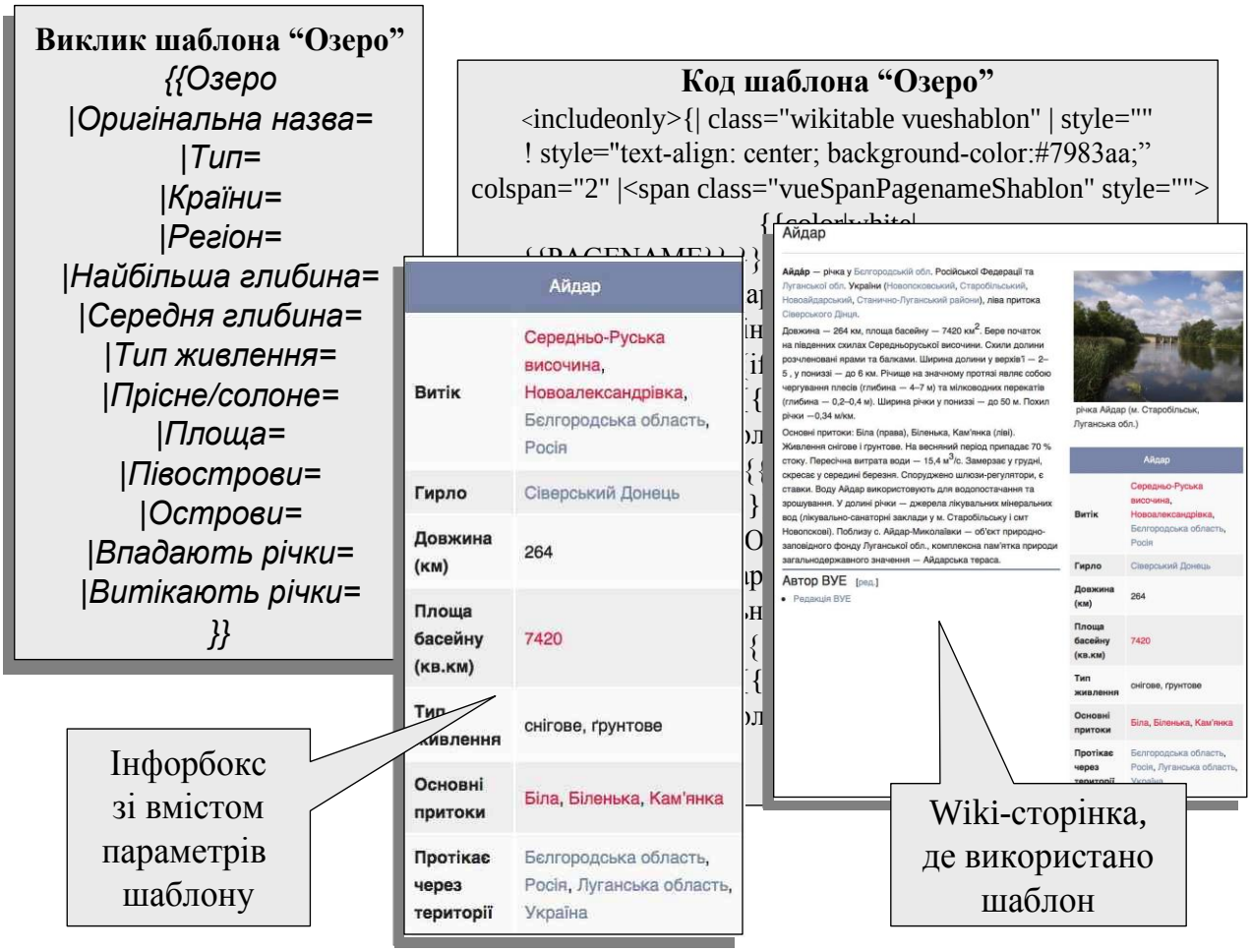


Рис. 2. Шаблони базових ІО в е-ВУЕ

Виразні можливості Semantic MediaWiki дозволяють відображати досить складну систему знань предметної області та відношень між її базовими поняттями, але їх недостатньо для формального подання таких описів, що мали б однозначну інтерпретацію користувачами. Це викликає доцільність застосування більш виразної онтологічної моделі, яка забезпечує вищий рівень формалізації знань.

Wiki-онтологія e-BYE

Wiki-онтологія e-BYE як модель знань цієї предметної області дозволяє коректно відображати його базові закономірності та обмеження. Використання Wiki-онтології як основи семантичної розмітки підтримує формування відповідного набору ієрархічно пов'язаних категорій, шаблонів типових інформаційних об'єктів, їх семантичних властивостей та запитів, що їх використовують.

Використання Wiki-онтології e-BYE є засобом явного визначення семантики типових ІО Wiki-ресурсу – їх характеристик та відношень з іншими ІО. Важливо, що таке онтологічне подання дозволяє виявляти та вирішувати неоднозначні інтерпретації та некоректне використання термінів, пов'язаних з описом ІО. Наприклад, це дозволяє розв'язати проблему уніфікації назв семантичних властивостей та категорій, які використовують різні розробники.

Крім того, онтологічне подання спрощує сприйняття знань користувачами, наприклад, дозволяє візуалізувати ієрархічні відношення між категоріями $r_{ier\_categor}$, описувати їх характеристики та надавати анотації. Для складної структури знань, що характерна для e-BYE, це дозволяє значно чіткіше описувати знання та запобігати повторного використання імен категорій з різними значеннями. Редактор онтологій Protégé забезпечує значно ширший функціонал для подання та візуалізації знань порівняно із вбудованими можливостями Semantic MediaWiki (рис. 3).

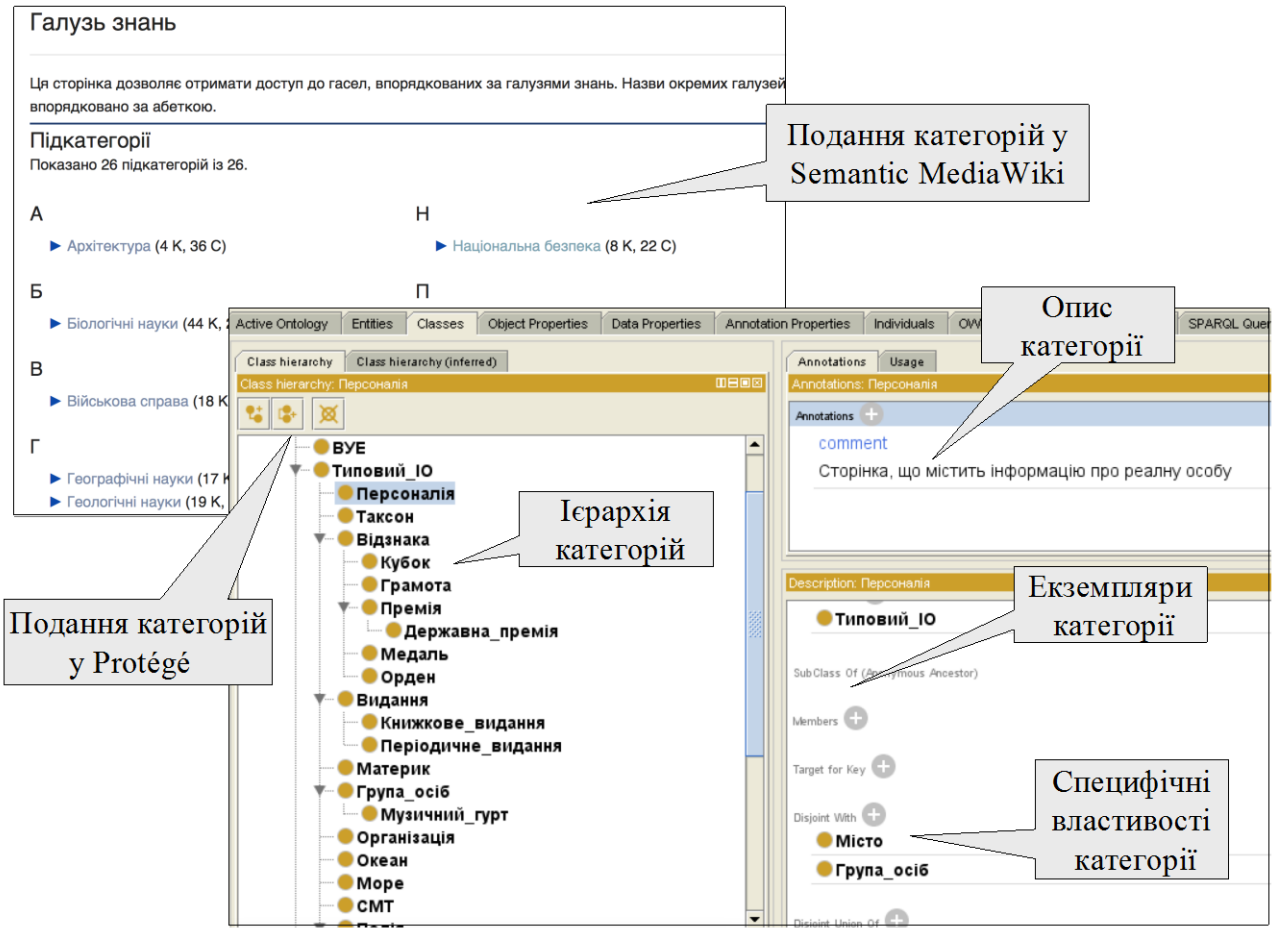


Рис. 3. Класи Wiki-онтології e-BYE в Protégé

Семантичні властивості в е-ВУЕ описуються у Wiki-онтології за допомогою об'єктних властивостей (для значень типу "Сторінка") та властивостей даних (для значень інших типів.

Особливо важливо формально описати семантичні властивості, що пов'язують Wiki-сторінки (в онтології для цього застосовуються об'єктні властивості). Наявність такого опису дозволяє полегшити семантичну розмітку природномовного тексту та перетворення НСД на структуровану базу знань. Використання онтологічної моделі дозволяє чітко визначити область значення та область визначення семантичних властивостей, які характеризують Wiki-сторінку, що пов'язана з певним типом інформаційного об'єкта за допомогою шаблону. Така формалізація семантики сторінок дозволяє запобігти некоректних посилань. Слід відмітити, що в Protege можна формально зафіксувати характеристики властивостей, щоб використовувати їх в е-ВУЕ тільки відповідним чином (рис. 4).

Редактор онтологій дозволяє явно вказати, що певна властивість є симетричною чи транзитивною. В е-ВУЕ такі характеристики має, наприклад, семантична властивість "Співпраця". Крім того, в Protege як об'єктні властивості, так і властивості даних створюються як підкатегорії інших властивостей, і ця ієрархія візуалізується у простій та зрозумілій формі. У середовищі Semantic MediaWiki можна також вказувати ієрархічні відношення між властивостями – за допомогою тверджень [[Subproperty of::dc:date]] на сторінці властивості, але відстежувати ці взаємини має сам користувач вручну. За наявності великої кількості як властивостей, так і користувачів, що мають право їх створювати (а саме така ситуація характерна для е-ВУЕ) це може призвести до некоректності у структурі бази знань енциклопедії.

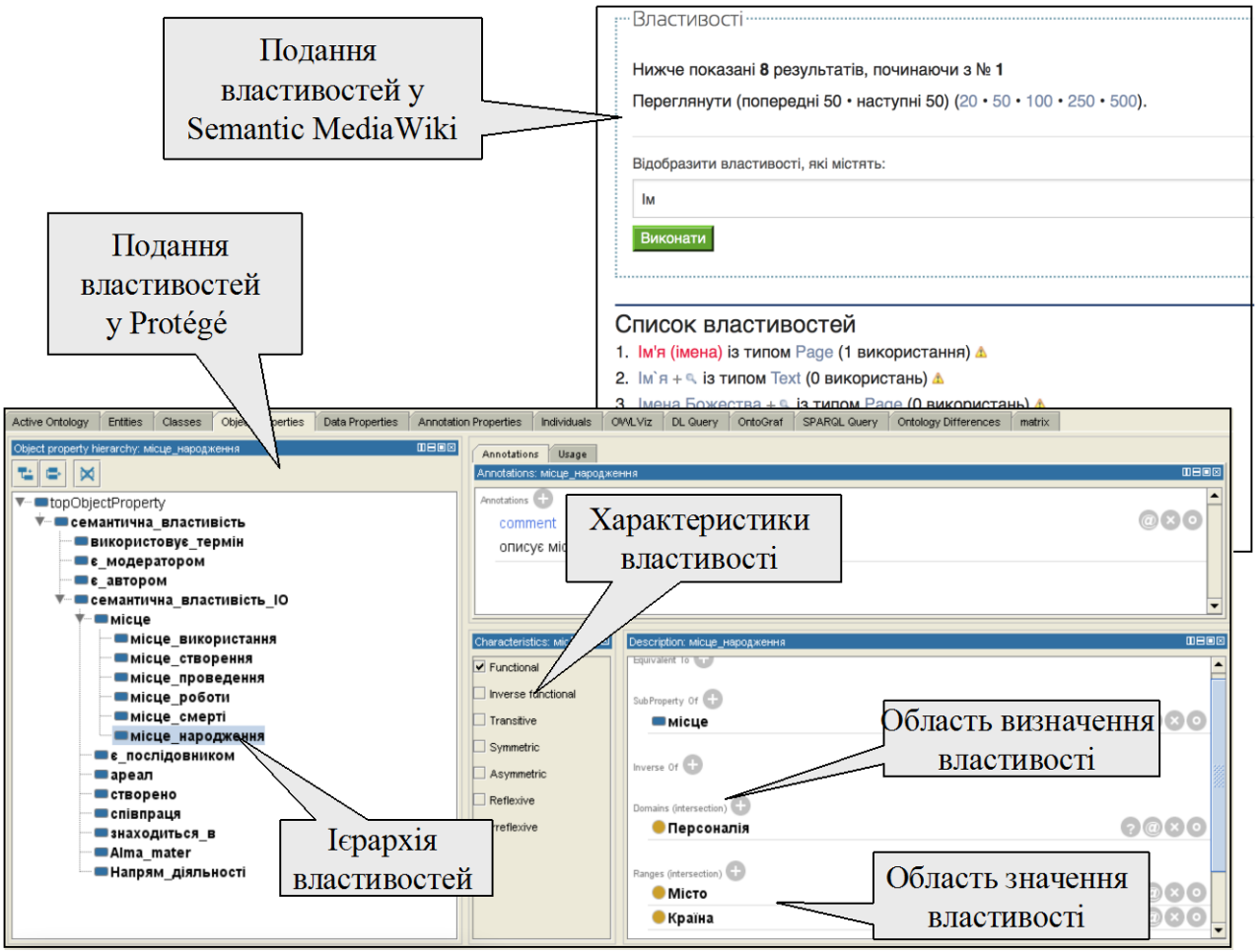


Рис. 4. Створення властивостей Wiki-онтології е-ВУЕ в Protégé

Важливо відмітити, що засоби Semantic MediaWiki дозволяють вказати тип властивості, поділяючи їх на об'єктні властивості, що посилаються на інші сторінки, та на властивості даних, що посилаються на конкретні значення (текст, ціле, дата тощо). Але це не дозволяє визначати тип інформаційного об'єкта, на який таке посилання є релевантним. Наприклад, за змістом "Місце народження" може пов'язувати сторінки категорії "Персоналія" зі сторінками категорій "Місто", "Країна", "Регіон", але формально визначити це засоби Semantic MediaWiki не дозволяють (можна тільки явно вказати припустимі значення, а цього недостатньо).

Wiki-онтологія дозволяє формалізувати набір властивостей кожної Wiki-сторінки як екземпляра відповідного класу.

Коли структура бази знань ресурсу містить багато різних типів інформаційних об'єктів з різними наборами властивостей, що перетинаються, але мають різні області значення та визначення, велику роль грає наявність засобів візуалізації цієї інформації у зручній для користувача формі. На жаль, середовище Semantic MediaWiki не може розглядатися як повноцінна система менеджменту розподілених знань: незважаючи на достатню для широкого класу задач виразність, вона не містить достатніх засобів для аналізу та візуалізації таких знань (інформація видається у формі абеткового переліку, через набори властивостей окремих Wiki-сторінок або як результати запитів тощо, але її важко сприймати у такому поданні).

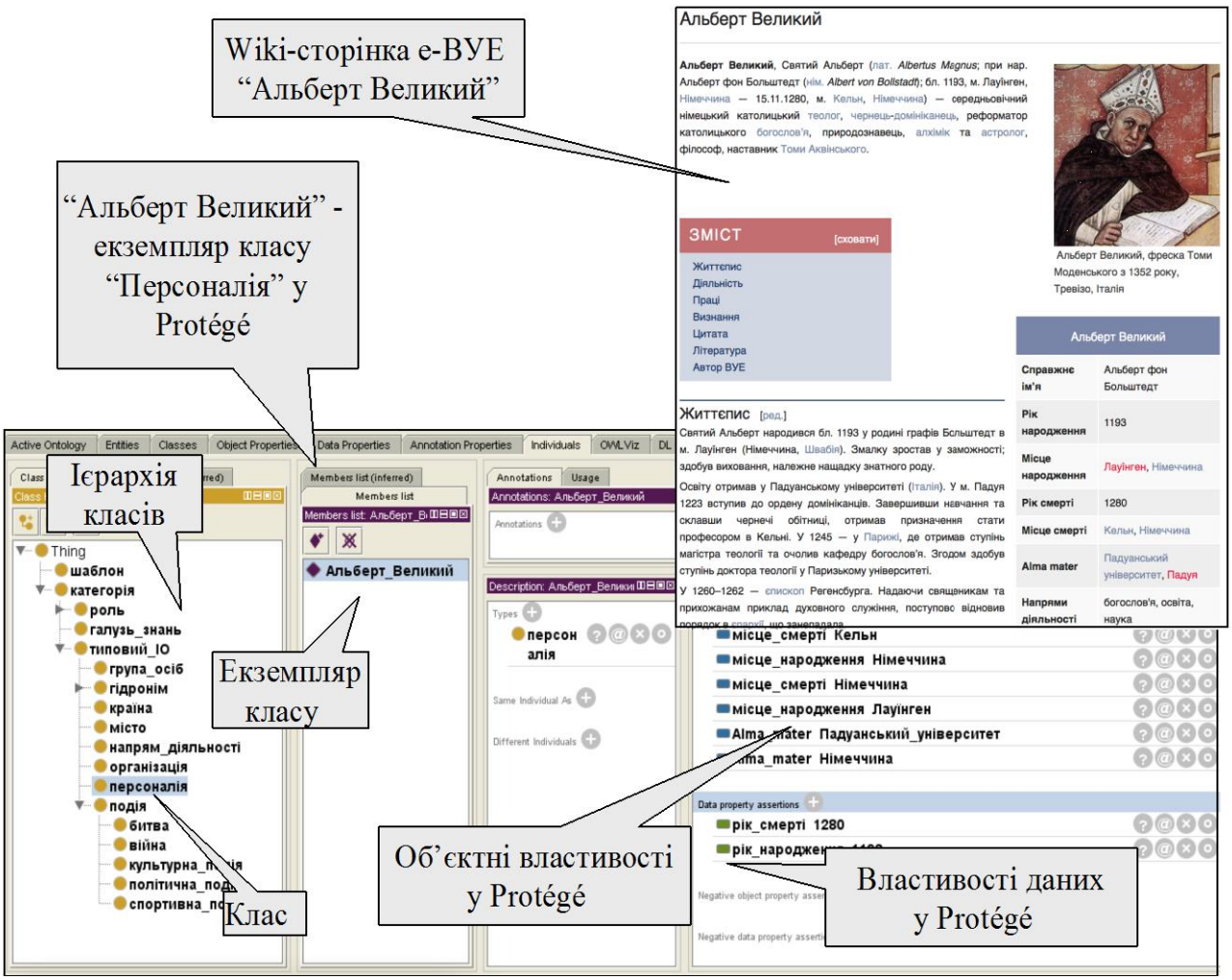


Рис. 5. Подання знань щодо ІО у e-BYE та у Wiki-онтології

Висновки

Проаналізовано доцільність застосування Wiki-технологій для створення складних інформаційних ресурсів великого обсягу та складної структури, орієнтованих на функціювання у відкритому середовищі Web, до яких відноситься портальна версія Великої української енциклопедії.

Обгрунтована потреба семантичного розширення Wiki-технології, а саме – використання Semantic MediaWiki, для отримання необхідної виразної потужності для подання семантики основних елементів е-ВУЕ, їх властивостей та зв'язків між ними. Розглянуто зв'язок між поданням знань у Semantic MediaWiki та стандартами Semantic Web, які використовуються для інтеграції інтелектуальних застосунків та баз знань у Web.

Доведено на прикладах доцільність застосування онтологічного аналізу та побудови онтологічної моделі е-ВУЕ для формального та однозначного відображення та аналізу структури та контенту бази знань онлайн-версії «Великої української енциклопедії».

Запропоновано методи та засоби застосування цієї онтологічної моделі для створення контенту е-ВУЕ та більш ефективного здійснення пошуку та навігації у цьому інформаційному ресурсі.

Література

1. Breslin J., Passant A., Deker S. The Social Semantic Web, Springer. 2009. http://staff.um.edu.mt/cabe2/lectures/webscience/docs/S2N.pdf.
2. Hagedorn, G., Weber, G., Plank, A. Giurgiu, M. Homodi, A., C. Veja An online authoring and publishing platform for field guides and identification tools. Proc. of Conf. Tools for Identifying Biodiversity: Progress and Problems, Paris, September 2010. P. 13–18.
3. Leuf B., Cunningham W. The Wiki Way Quick Collaboration on the Web. 2001. https://archive.org/details/isbn_9780201714999.
4. Рогушина Ю. В. Засоби та методи аналізу неструктурованих даних. *Проблеми програмування.* 2019. № 1. С. 57–77. http://pp.isofts.kiev.ua/ojs1/article/view/348/346.
5. Veja C.F., Vaida M.-F., Hagedorn G. Sharing the knowledge: semantic mediawiki. *Acta Technica Napocensis*. 2011. 52(2), N.2. http://users.utcluj.ro/~atn/papers/ATN_2_2011_2.pdf.
6. Рогушина Ю.В., Прийма С.М., Строкань О.В. Створення та використання семантичних Wiki-ресурсів: навчальний довідник. Мелітополь, ФОП Однорог Т.В. 2017. 169 с.
7. Kuhn T. Acewiki: A natural and expressive semantic wiki. 2008. https://arxiv.org/pdf/0807.4618.pdf.
8. Schaffert S., Eder J., Grünwald S., Kurz T., Radulescu M. KiWi – a platform for semantic social software. European Semantic Web Conference. 2009. P. 888–892. https://link.springer.com/content/pdf/10.1007/978-3-642-02121-3_76.pdf.). Springer, Berlin, Heidelberg.
9. Kawamoto K., Kitamura Y., Tijerino Y. Kawawiki: A semantic wiki based on rdf templates. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops. 2006. P. 425–432. https://ist.ksc.kwansei.ac.jp/~kitamura/papers/src/iwi06.pdf.
10. Auer S., Dietzold S., Riechert T. OntoWiki–a tool for social, semantic collaboration. International Semantic Web Conference. 2006. P. 736–749. https://link.springer.com/content/pdf/10.1007/11926078_53.pdf.
11. Krötzsch M., Vrandečić D., Völkel M. Semantic mediawiki. International semantic web conference. 2006. P. 935–942. https://link.springer.com/content/pdf/10.1007/11926078_68.pdf.
12. Kousetti Ch., Millard I., Yvonne H. A Study of Ontology Convergence in a Semantic Wiki. Based Infrastructure. University of Southampton. UK, 2008.
13. Bizer C., Lehmann J., Kobilarov G., Auer S., Becker C., Cyganiak R., Hellmann S. Dbpedia – A crystallization point for the Web of Data. Web Semantics: science, services and agents on the world wide web. 2009. 7(3). P. 154–165. https://gesispanel.gesis.org/preprints/index.php/ps/article/download/164/162.
14. Auer S., Lehmann J. What have innsbruck and leipzig in common? extracting semantics from wiki content. European semantic web conference. 2007. P. 503–517. https://link.springer.com/content/pdf/10.1007/978-3-540-72667-8_36.pdf.
15. Rogushina J. Semantic Wiki resources and their use for the construction of personalized

ontologies. CEUR Workshop Proceedings 1631. 2016. P. 188–195.
16. Рогушина Ю.В. Теоретичні засади застосування онтологій для семантизації ресурсів Web. *Проблеми програмування*. 2018. № 2-3. С. 197–203.

References

1. Breslin J., Passant A., Deker S. The Social Semantic Web, Springer, 2009. http://staff.um.edu.mt/cabe2/lectures/webscience/docs/S2N.pdf.
2. Hagedorn, G., Weber, G., Plank, A. Giurgiu, M. Homodi, A., C. Veja An online authoring and publishing platform for field guides and identification tools // Proc. of Conf. Tools for Identifying Biodiversity: Progress and Problems, Paris, September 2010. P. 13–18.
3. Leuf B., Cunningham W. The Wiki Way Quick Collaboration on the Web, 2001. https://archive.org/details/isbn_9780201714999.
4. Rogushina Y.V. Tools and methods of unstructured data analysis // Problems in Programming, № 1, 2019. P. 57–77. http://pp.isofts.kiev.ua/ojs1/article/view/348/346. [in Ukrainian]
5. Veja C.F., Vaida M.-F., Hagedorn G. Sharing the knowledge: semantic mediawiki // Acta Technica Napocensis, 52(2), N. 2, 2011. – http://users.utcluj.ro/~atn/papers/ATN_2_2011_2.pdf.
6. Rogushina Y.V., Priyma S.M, Strokan O.V. Creating and use of the Semantic Wiki resources: tutorial. Melitopol, FOP Odinorog T.V., 2017. 169 p. [in Ukrainian]
7. Kuhn T. Acewiki: A natural and expressive semantic wiki. 2008. https://arxiv.org/pdf/0807.4618.pdf.
8. Schaffert S., Eder J., Grünwald S., Kurz T., Radulescu M. KiWi – a platform for semantic social software. European Semantic Web Conference, 2009. P. 888–892. https://link.springer.com/content/pdf/10.1007/978-3-642-02121-3_76.pdf.). Springer, Berlin, Heidelberg.
9. Kawamoto K., Kitamura Y., Tijerino Y. Kawawiki: A semantic wiki based on rdf templates. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, 2006, P. 425–432. https://ist.ksc.kwansei.ac.jp/~kitamura/papers/src/iwi06.pdf.
10. Auer S., Dietzold S., Riechert T. OntoWiki–a tool for social, semantic collaboration // International Semantic Web Conference, 2006, P. 736–749. https://link.springer.com/content/pdf/10.1007/11926078_53.pdf.
11. Krötzsch M., Vrandečić D., Völkel M. Semantic mediawiki. International semantic web conference. 2006. P. 935–942. https://link.springer.com/content/pdf/10.1007/11926078_68.pdf.
12. Kousetti Ch., Millard I., Yvonne H. A Study of Ontology Convergence in a Semantic Wiki. Based Infrastructure. University of Southampton. UK, 2008.
13. Bizer C., Lehmann J., Kobilarov G., Auer S., Becker C., Cyganiak R., Hellmann S. Dbpedia – A crystallization point for the Web of Data. Web Semantics: science, services and agents on the world wide web. 2009. 7(3). P. 154–165. https://gesispanel.gesis.org/preprints/index.php/ps/article/download/164/162.
14. Auer S., Lehmann J. What have innsbruck and leipzig in common? extracting semantics from wiki content. European semantic web conference. 2007. P. 503–517. https://link.springer.com/content/pdf/10.1007/978-3-540-72667-8_36.pdf.
15. Rogushina J. Semantic Wiki resources and their use for the construction of personalized ontologies. CEUR Workshop Proceedings 1631. 2016. P. 188–195.
16. Rogushina Y.V. Theoretical means of use of ontologies for semanization of the Web resources. Problems in Programming. № 2-3. 2018. P. 197–203. [in Ukrainian]

Одержано 22.04.2019

Про автора:

Рогушина Юлія Віталіївна,
кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 140.
Кількість наукових публікацій в зарубіжних виданнях – 30.
Індекс Хірша – 3.
http://orcid.org/0000-0001-7958-2557.

Місце роботи автора:

Інститут програмних систем
НАН України,
03181, Київ-187,
проспект Академіка Глушкова, 40.
Тел.: 066 550 1999.
E-mail: ladamandraka2010@gmail.com

УДК 004.82 https://doi.org/10.15407/pp2019.02.038

К.А. Кудим, Г.Ю. Проскудина

МЕТОДЫ И СРЕДСТВА ИЗВЛЕЧЕНИЯ ДАННЫХ О ПЕРСОНАЛИЯХ ИЗ АВТОРЕФЕРАТОВ ДИССЕРТАЦИЙ

В работе рассмотрены подходы к решению задачи сбора и извлечения разрозненных данных о персоналиях из слабоструктурированных и неструктурированных документов, представленных в общедоступных каталогах авторефератов диссертаций. На языке PHP с применением XPath разработана система, которая позволяет автоматически собирать первичные документы из электронного каталога Национальной библиотеки Украины им. В.И. Вернадского, извлекать из этих документов данные и сохранять их в локальном хранилище. Для хранения выбрана модель данных RDF с учётом особенностей данных и возможностью последующего представления в семантической сети.

Ключевые слова: извлечение данных, слабоструктурированные документы, технология XPath, регулярные выражения, семантическая сеть.

Введение

В настоящее время в Интернете доступен большой объём разрозненной публичной информации о персоналиях, не представленной явно, а содержащийся в документах и электронных каталогах как дополнительные сведения. Например, сведения об учёных защитивших диссертацию могут быть найдены в авторефератах диссертаций.

В собранном виде эти данные могут представлять ценность как независимый ресурс. Однако сбор таких данных затруднён как из-за количества информации для обработки, так и по причине её слабой структурированности. Учитывая данное обстоятельство предлагается автоматизировать сбор и извлечение фактов о персоналиях из общедоступных источников, в частности из авторефератов диссертаций.

В качестве ресурса для реализации прототипа был выбран электронный каталог авторефератов диссертаций Национальной библиотеки Украины им. В.И. Вернадского.

1. Постановка задачи

Некоторое время, занимаясь созданием электронной большой Украинской Энциклопедии, в частности, Википедии и семантической Вики, мы обнаружили проект Викидата [1] и начали подробно изучать этот проект.

Что это такое в целом? Известно, что Википедия представляет собой большую коллекцию статей, которые наполняются пользователями в довольно произвольной форме. В настоящее время существует параллельный проект Викидата, где реализована попытка представить информацию Википедии только исключительно в структурированном виде. Причем так, чтобы ее можно было обрабатывать как машиной, так и человеком.

Почему такая проблема возникла? Дело в том, что в Википедии есть много языковых разделов (несколько десятков), и некоторая информация дублируется в каждом языковом разделе абсолютно идентично. Например, у каждой статьи есть список ссылок на ту же статью на других языках. И этот список в каждом языковом разделе будет один и тот же. Но в каждом языковом разделе, будут другие пользователи, которые редактируют эти списки. Так в Википедии появились боты, которые сами обновляют эти списки. А в проекте Викидата все решается просто – есть одно централизованное хранилище, которые хранит общие данные, а каждая языковая Википедия может вставлять эти данные на свои страницы. Можно сказать, что Викидата это база данных, где хранятся не просто статьи, а структурированные данные. Причем эти данные настолько подробные, что из них можно генерировать статьи, подобные википедийным.

На рис. 1 приведен пример статьи, которая сгенерирована по данным [2]. Это не статья, которую набирал человек. При

этом ее сделали человеко-читаемой. Здесь есть краткая биография И.С. Баха, когда родился, когда умер, основные факты из биографии. Далее перечислены родители, дети. Весь этот сгенерированный контент берется из базы данных Викидата. Здесь также есть аудио, видео и графический контент, относящийся в биографии этого композитора, который берется из некоторых других централизованных источников информации соответствующего типа.

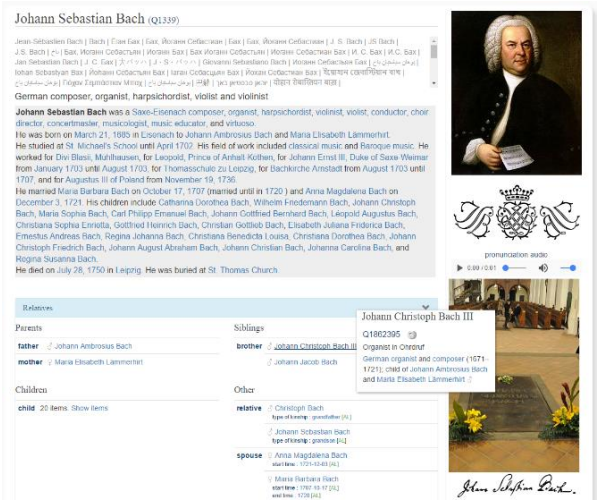


Рис. 1. Автоматически сгенерированная страница о И.С. Бахе

Викидата это – отдельный централизованный проект со своей администрацией и своими правилами. Например, на него распространяются правила википедийной *значимости*. Нельзя заносить в базу данных Викидата любые объекты подряд. Например, мы не можем заносить туда своих кандидатов наук, если это по мнению администраторов Викидата не выдающиеся ученые. Или какие-то значимые для нас статьи, которые не значимые для Википедии, также туда попасть не могут. Это первый момент, второй – это доступ. Для того, чтобы полноценно использовать базу Викидата, предлагается ее полностью скачать (а база данных Викидата предоставляет такую возможность) и установить себе. То есть централизованно базу данных Викидата использовать не эффективно.

Мы также задались вопросом, а можем ли мы найти в Интернете список всех ученых Украины, например, кандидатов и докторов наук? В настоящее время таких списков, централизованных нет, по крайней мере, мы не смогли их найти. И так, чтобы они были общедоступными. Некоторые ресурсы предоставляют списки докторов наук, а списков кандидатов наук вообще нет.

На основании вышеизложенного, по аналогии с проектом Викидата, принято решение о создании собственной централизованной базы данных Персоналии, руководствуясь следующими принципами:

- максимальное включение;
- публичные данные;
- источники информации: открытые, авторитетные и проверяемые;
- используется структурированное хранение (например, в формате RDF).

Таким образом в базу Персоналии, можно включать не только значимых персон, а *всех* персон, о которых есть публичная информация в Интернете, не нарушив ее приватности. Это могут быть ученые, кандидаты и доктора наук, это могут быть чиновники, врачи, учителя, сотрудники различных бюджетных организаций. Сюда можно добавлять всех, о ком, что-либо известно.

В Википедии считают, что, например, географический объект имеет право быть включенным в Википедию. Людей это не касается. И нередко в Википедии случаются диспуты по поводу персон, которые кому-то кажутся значимыми и их хотят включить в Википедию, а другой части сообщества эти персоны кажутся не значимыми, и они стараются удалять статьи о таких персонах. Нередко ведутся целые споры и пожизненные блокировки. Как говорят в Википедии, используется *имманентная значимость*.

Источники кого-то включить в базу данных Персоналии должны быть, как и в Википедии *авторитетными*, нельзя со своих слов кого-то включить. Обязательно должно быть подтверждение, и это должно быть доступное подтверждение. Информация должна размещаться на авторитетном ресурсе, принадлежащем организации к которой есть доверие всех участников.

Предполагается, что в базу данных Персоналии будет вноситься открытая и

достоверная информация обо всех людях Украины, но на первом этапе, ограничиться учеными Украины.

В итоге принято решение сохранить преимущество базы данных Викидата, чтобы хранилище оставалось структурированным, а не в виде хранилища документов. Поскольку к структурированному хранилищу можно осуществлять запросы.

2. Сбор данных

Далее возник вопрос наполнения. Естественно вручную не реально наполнять такую базу данных. В этом мы убедились, наполняя статьями нашу Библиотеку Периодических изданий[1]. Ручной ввод информации – весьма трудозатратное занятие. Даже когда доступны все материалы, введение в электронную библиотеку одного элемента данных занимает некоторое время (3–5 минут), умножая на то количество данных, которое надо ввести, получаем около тысячи часов. Поэтому нас будет интересовать именно *автоматическое наполнение*.

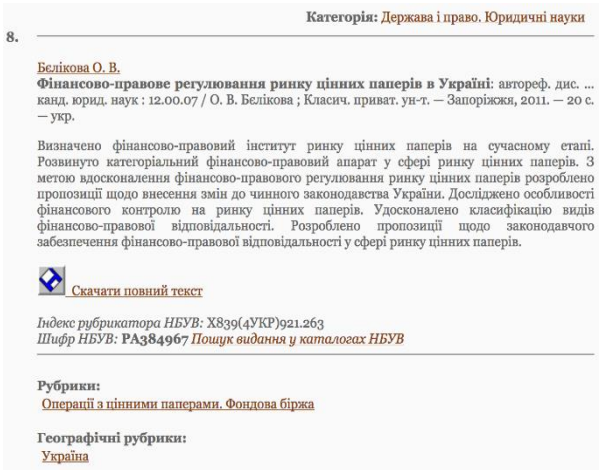
В качестве авторитетного источника (или исходных) данных был выбран сайт Национальной библиотеки Украины им. В.И. Вернадского, в части хранения авторефератов диссертаций[2]. На сегодня предоставляется открытый доступ к авторефератам всех диссертаций, защищенных в Украине за период 1998–2011 гг. Представлены они довольно просто: есть каталог и есть полные тексты авторефератов. Там имеется более 63 тыс. документов. И есть в каталоге некоторые об этих документах сведения, которые можно получить, не анализируя сами документы.

Осуществив навигацию по интерфейсу сайта Библиотеки им. В.И. Вернадского, было обнаружено, что у них есть навигация новых поступлений (рис. 2). Можно выбрать год и месяц и получить поступления не на этот месяц и год, а просто закачанные в этот месяц и год. Но для нас это неважно, главное, что мы получаем полный список авторефератов диссертаций.

Рис. 2. Навигация новых поступлений на сайте Национальной библиотеки Украины им. В.И. Вернадского

Был написан специальный скрипт, который эту навигацию делает автоматически. Начиная с 2007 года, когда они начали закачивать файлы, по 2012 год, обходит все месяцы. Таким образом были получены результаты поиска в html-формате, из которых следует уже извлекать первые данные.

На рис. 3. показано, как выглядит один элемент результата поиска. Это некоторая информация об автореферате диссертации.


Категорія: Держава і право. Юридичні науки

8.

Белікова О. В.
Фінансово-правове регулювання ринку цінних паперів в Україні: автореф. дис. ... канд. юрид. наук : 12.00.07 / О. В. Белікова ; Класич. приват. ун-т. — Запоріжжя, 2011. — 20 с. — укр.

Визначено фінансово-правовий інститут ринку цінних паперів на сучасному етапі. Розвинуто категоріальний фінансово-правовий апарат у сфері ринку цінних паперів. З метою вдосконалення фінансово-правового регулювання ринку цінних паперів розроблено пропозиції щодо внесення змін до чинного законодавства України. Досліджено особливості фінансового контролю на ринку цінних паперів. Удосконалено класифікацію видів фінансово-правової відповідальності. Розроблено пропозиції щодо законодавчого забезпечення фінансово-правової відповідальності у сфері ринку цінних паперів.

Скачати повний текст

Індекс рубрикатора НБУВ: Х839(4УКР)921.263
Шифр НБУВ: **РА384967** *Пошук видання у каталогах НБУВ*

Рубрики:
Операції з цінними паперами. Фондова біржа

Географічні рубрики:
Україна

Рис. 3. Библиографическая карточка автореферата диссертации

Для сбора первичных данных из электронного каталога используется поисковый робот (веб-краулер), который обходит страницы по навигационным ссылкам и сохраняет доступную информацию в локальном хранилище в необработанном виде. Поскольку поисковый робот не является универсальным и выборочно сохраняет

[1] http://dspace.nbuv.gov.ua
[2] http://nbuv.gov.ua/node/1539

только необходимую для последующей обработки информацию, то для каждого электронного каталога (источника данных) алгоритм обхода страниц по ссылкам должен быть адаптирован.

Локально копируется информация двух видов: HTML страницы и полные тексты авторефератов. Сохраненные HTML страницы используются поисковым роботом для дальнейшей навигации по страницам, а также для дальнейшей обработки, поскольку содержат также полезную информацию о документах. Полные тексты представлены файлами различных форматов, таких как PDF, RTF, DOC, которые сохраняются в оригинальном виде для последующей обработки.

После того, как все страницы результатов поиска были сохранены, были также закачаны все полные тексты. Их оказалось довольно много, как уже говорилось выше, более 63 тыс. авторефератов, что в общей сложности заняло 8.5 Гб.

Поисковый робот написан на PHP. Чтобы облегчить задачу адаптации программы под конкретный каталог, все вспомогательные функции вынесены из основной части. Обнаружение и извлечение ссылок для перехода на следующие страницы осуществляется с помощью языка XPath.

XPath является лаконичным языком для поиска узлов в XML документах, в частности им можно обрабатывать HTML страницы. Его использование позволяет сократить затраты на адаптацию алгоритма под разные ресурсы.

3. Обработка HTML страниц

Поскольку страницы, такие как html или xml, можно отнести к слабоструктурированным данным, где нет строгих типов, как в реляционных базах, и дерево может ветвиться очень по-разному. Извлечение нужных данных из корпуса таких страниц может осуществляться разными способами:

- с помощью регулярных выражений[3];
- навигацией по объектной модели документа (DOM);
- с помощью языка XPath;
- с использованием селекторов CSS.

Остановимся на них несколько подробнее. Как известно, регулярные выражения используются для формализации поисковых запросов в тексте. Регулярные выражения обладают известными недостатками. При достаточной сложности регулярного выражения (а при парсинге html возникают достаточно сложные регулярные выражения), они становятся очень трудно поддерживаемыми. Если страничка незначительно меняется, регулярное выражение приходится чуть ли не полностью переписывать. Их трудно понять другому человеку, не тому, кто их писал. И самое главное – они игнорируют структуру html, и рассматривают документ, просто как плоский текст. А html – структурированный текст, и почему бы этим не воспользоваться.

Есть и другая возможность разбора html, используя объектную модель документа, как это делают браузеры, которые используют структуру DOM для отображения документа. То есть на языке программирования входить в эту структуру и выбирать нужные узлы. Недостаток такого способа извлечения данных – слишком громоздко и зависимо от языка. Это также трудно поддерживать, поскольку опять же, если структура странички изменится, то придется переделывать и код программы.

Поэтому для решения поставленной задачи наиболее предпочтительным оказался язык для поиска выражений в xml-структурах. Называется этот язык XPath (икс-путь). Расшифровывается как xml path language, т. е. язык xml-путей. Он используется во многих стандартах xml. В отличие от регулярных выражений он позволяет писать запросы не как к плоскому тексту, а как к дереву. При этом запросы получаются достаточно простыми. Подмножества этого языка – селекторы CSS. Это тоже удобный инструмент, еще проще, чем язык XPath, но XPath – мощнее. Поэтому далее более подробно рассмотрим этот язык.

[3] Язык регулярных выражений, основанный на автоматах, для поиска подстрок в тексте.

Как уже было сказано выше, XPath – язык для навигации по xml-документу. Если в xml-документе необходимо найти какой-нибудь узел, то это можно сделать, либо обходом xml-дерева, либо просто воспользоваться готовым языком. Он позволяет небольшим выражением найти все узлы, которые соответствуют определенным критериям: по имени, по атрибуту узла и по другим условиям, по которым нам хотелось бы искать эти узлы.

XPath – весьма популярный язык, является стандартом консорциума W3C с 1999 года. Он лежит в основе многих других стандартов XML (рис. 4). Такие, например, как XQuery – очень мощный язык не только запросов к большим коллекциям xml-документов, но и есть полноценным языком программирования. В последние годы наблюдается перенос XQuery в языки программирования, на нем можно писать даже сайты, как на PHP, только в основе лежит XML.

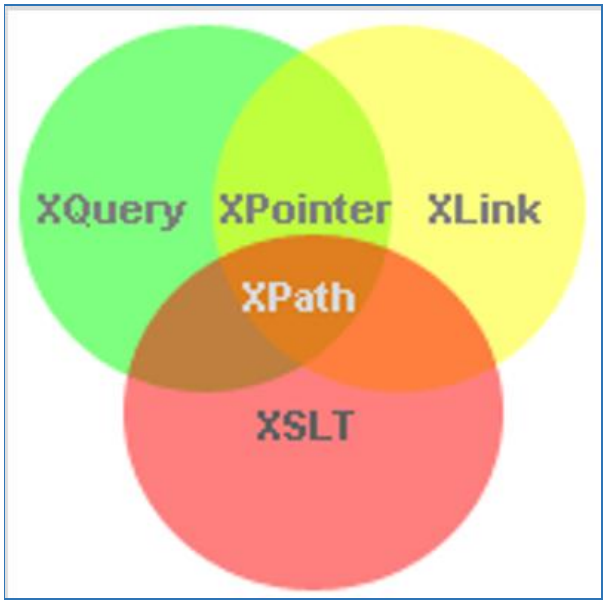


Рис. 4. Язык XPath как основа других XML стандартов консорциума W3C

Выражения XPath представляют собой некоторый путь. Например, на рис. 5. показаны html-узлы, где путь в корне *tr/td/a* – строка таблицы, потом ячейка таблицы, потом ссылка. То есть для того чтобы найти все ссылки документа, которые лежат в ячейках таблицы, можно написать такой простой запрос. Для сравнения можно представить, сколько бы занимал обход дерева на каком-нибудь языке программирования или каким бы регулярным выражением пришлось бы отыскивать все такие ссылки.

- Путь — /tr/td/a
  - Шаги — tr
    - Ось — **child::**tr
    - Проверка узла — child::**tr**
    - Предикат(ы) — [text()='a']
- Функции — position()
- Операции — |, //
- Контекст

Рис. 5. Основы языка XPath

Путь разбивается на шаги, каждый шаг представляет собой некий узел в дереве xml-разбора (в нашем случае html-разбора). Каждый шаг делится на ось, по которой происходит поиск. Ось это – либо дочерние узлы, либо родительские узлы, либо атрибуты, либо соседние с данным узлом узлы. Вторая часть шага – проверка узла, т. е. осуществляется поиск всех узлов, совпадающие с *tr*, строкой таблицы. В итоге третья часть шага – один или несколько предикатов, которые еще больше сужают область поиска. Например, если внутри узла у нас должен содержаться текст ‘a’, то можно написать следующий предикат: [text()=’a’], дополнительно фильтрующий множество найденных узлов.

Также в XPath определено много дополнительных функций, которые позволяют вычислять позицию этого узла, а также есть много строковых и арифметических функций. Есть также операции, которые позволяют объединять результаты, пропускать узлы в дереве, вплоть до дочерних. И что важно, XPath осуществляет поиск, начиная не обязательно от корня, а от любого узла в дереве, если ему задан контекст, в котором искать.

С помощью этих не хитрых возможностей можно делать весьма сложные запросы к html-документам. Вот несколько примеров. На рис. 6 приведены фрагменты

скрипта, который выбирает данные из html-страницы результатов поиска.

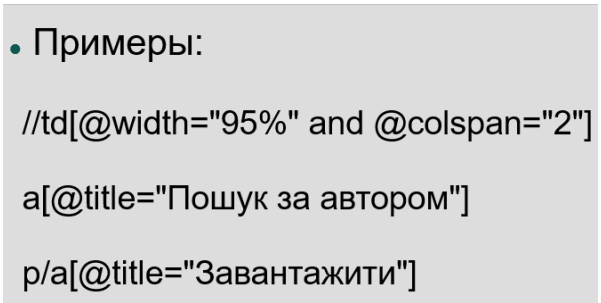


Рис. 6. Примеры поисковых выражений XPath

Вначале мы ищем все узлы на любом уровне дерева, которые являются ячейкой данных таблицы с атрибутами *width="95%" and* (здесь логическое “и”) *colspan="2"*. Так мы находим все ячейки в таблице, где находится один результат поиска. Дальше, используя этот узел как контекстный узел, мы ищем имя автора следующим образом. Находим ссылку (a) с атрибутом *title="Пошук за автором"* и таким образом найдем фамилию и инициалы автора. Если же мы найдем ссылку (a), которая является дочерним узлом параграфа (p) и имеет атрибут *title="Завантажити*", то мы найдем ссылку на полный текст.

Таким образом, сохранённые в локальном хранилище HTML документы из каталога электронной библиотеки уже содержат некоторую информацию о диссертациях: имя автора, название, научную степень. Эту информацию достаточно легко извлечь, используя XPath.

Для этого на языке PHP была написана программа, которая извлекает нужные данные из HTML страницы и сохраняет их в базу данных.

4. Обработка неструктурированных документов

Полные тексты авторефератов представлены в различных форматах, таких как RTF, PDF, DOC, DOCX (рис. 7). Поскольку в этих форматах зафиксировано форматирование и не хранится семантическая информация, то с ними можно работать только после извлечения текстовых данных. Для каждого формата необходима своя программа, преобразующая входной файл в текстовый формат.

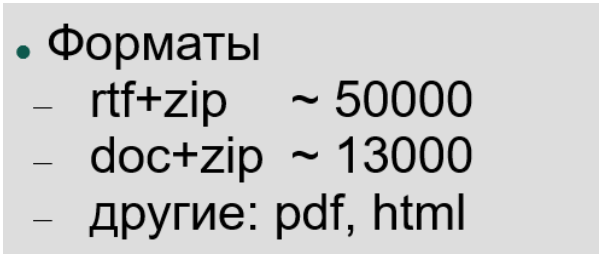


Рис. 7. Корпус текстов авторефератов диссертаций с точки зрения их форматов

Поскольку большинство файлов выбранного библиотечного ресурса представлено в формате RTF, в первую очередь была разработана программа для преобразования этого формата в обычный текст. И следующий модуль с помощью регулярных выражений извлекает из текста целевые сущности.

Формат RTF – довольно простой. Для работы с ним был взят готовый парсер, с последующей его доработкой. Преимуществом данного формата для нас явилось то, что внутреннее его представление не бинарное, а в символах ASCII. Вся структура документа состоит из двух сущностей: это – *управляющие слова*, которые начинаются с обратного слэша и *группы*, которые получаются обрамлением фигурными скобками. Благодаря группам, удалось сразу удалить много “лишнего” из этого формата: всякие бинарные данные, картинки, шрифты, палитры цветов. Все остальное, что не является управляющими словами и группами, это символы. В случае кириллицы мы там символов не увидим, поскольку кириллические символы являются управляющими словами, и тут еще возникает задача перекодировки символов. То есть готовый парсинг нуждался в усовершенствовании, для того чтобы не приходилось после него еще и преобразовывать кодировку. После rtf-парсинга на выходе получаем простой неразмеченный текст.

После преобразования текст поступает на вход алгоритму, распознающему сущности, представляющие интерес для целей системы: имена научных руководителей, имена оппонентов, научные степе-

ни, названия организаций, в которых проводилась подготовка и защита диссертации, дата защиты, номер специальности. Всё, что пока удалось извлечь, сохраняется в базе данных.

Здесь еще предстоит дальнейшая работа над rtf-документом по извлечению именованных сущностей из плоского текста без структуры. Здесь нужно отметить, что решение этой задачи находится на начальной стадии. В настоящее время мы извлекаем из плоского текста регулярными выражениями, используя некоторые признаки. Например, порядок следования слов, фамилия обычно идет с инициалами или три слова с заглавными буквами; заголовок – все заглавные буквы; специальность – последовательность чисел с точками; для УДК тоже есть свой символьный формат или шаблон и т. д.

Пример регулярного выражения, извлекающего из плоского текста, имя автора, название диссертации, УДК и специальность:

/^.*?([$upper]+[$lower]* [$upper]+[$lower]* [$upper]+[$lower]*) ([Уу][ ]?[Дд][ ]?[Кк][:]?[ ]?[-–0-9.'“”»«:;\/\+\[\]\(\)ЄС]+) (.*?) ([0-9][ ]?[0-9][ ]?\.[ ]?[0-9][ ]?[0-9][ ]?\.[ ]?[0-9][ ]?[0-9][.]?[-]?) /

Стартовая программа прошла тестирование вначале на одном автореферате, а затем регулярные выражения были настроены на большее число авторефератов (в экспериментальном корпусе их было 300, выборка за месяц).

На рис. 8 показан фрагмент таблицы базы данных, где можно увидеть

yearmonth	id	author	name	link	fio	udc	nazvanie	speciality
201002	1	Дідора В.Г.	Агроекологічне обгрунтування технології виробництв...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	ВІ КТОР ГРИГОРОВИЧ	УДК 633.521:581.5/47	АГРОЕКОЛОГІ ЧНЕ ОБГРУНТУВАННЯ ТЕХНОЛОГІЇ ВИРО БН И...	06.01.09
201002	2	Сердюченко О.В.	Адміністративно-правові засади забезпечення енерге...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	ЕРДЮЧЕНКО Олексій Володимирович	УДК 347:620.9(477)	АДМІНІСТРАТИВНО-ПРАВОВІ ЗАСАДИ ЗАБЕЗПЕЧЕННЯ ЕНЕР Г...	12.00.07
201002	3	Гридасов Ю.В.	Адміністративно-правові засоби захисту права власн...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	ГРИДАСОВ ЮРІЙ ВОЛОДИМИРОВИЧ	УДК 342.951	АДМІНІСТРАТИВНО-ПРАВОВІ ЗАСОБИ ЗАХИСТУ ПРАВА ВЛАСН...	12.00.07
201002	4	Фурман О.А.	Активізація навчально-пізнавальної діяльності майб...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	УРМАН Олена Андріївна	УДК 378.851	АКТИВІЗАЦІЯ НАВЧАЛЬНО-ПІЗНАВАЛЬНОЇ ДІЯЛЬНОСТІ МАЙБ...	13.00.02
201002	5	Гаврилків В.М.	Алгебро-топологічні структури на суперрозширеннях	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	Гаврилків Володимир Михайлович	УДК 51 2.53	АЛГЕБРО-ТОПОЛОГІЧНІ СТРУКТУРИ НА СУПЕРРОЗШИРЕННЯХ	01.01.0 6
201002	7	Репетєва О.В.	Антигіпоксичні, антиоксидантні та антиексудативні ...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	РЕПЕТЄВА ОЛЕНА ВАЛЕРІЇВНА	УДК : 615.275.4:547.	АНТИГІПОКСИЧНІ, А НТИОКСИДАНТНІ ТА АНТИЕКСУДАТИВНІ...	14.03.05
201002	8	Нестерук С.В.	Багатоцільова антенна решітка з цифровим керування...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	Нестерук Сергій Володимирович	УДК 621.396.677	БАГАТОЦІЛЬОВА АНТЕННА РЕШІТКА З ЦИФРОВИМ КЕРУВАННЯ...	05.12.0 7
201002	9	Ткач Є.В.	Банківське кредитування інноваційного розвитку: вза... взаємозв'язок жанрів: новітн...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	Ткач Євген Вікторович Василівна	УДК 336.77:334.012.6 8	БАНКІВСЬКЕ КРЕДИТУВАННЯ ІННОВАЦІЙНОГО РОЗВИТКУ: ВЗАЄМОЗВ'ЯЗОК ЖАНРІВ: НОВІТН...	08.00.08
201002	14	Воліков І.О.	Вибір варіанта запобіжної аналгезії в абдомінальні...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	ВОЛІКОВ ІГОР ОЛЕКСАНДРОВИЧ	УДК: 617.55-089:616.	ВИБІР ВАРІАНТУ ЗА ПОБІЖНОЇ АНАЛГЕЗІЇ В АБДОМІНАЛЬН...	14.01.30
201002	15	Бегма Н.А.	Використання побічних продуктів переробки зерна ку...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	Бегма Наталія Андріївна	УДК 636.4.087.74	Використання побічних продуктів переробки зерна ку...	06.02.02
201002	16	Костащук В.І.	Використання та охорона мінерально-сировинних ресу...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	Костащук Володимир Іванович	УДК 911.3: 330.15 (4	В икористання та охорона мінерально-сировинних рес...	11.00. 02
201002	21	Василенко І.В.	Виховання громадянськості студентів засобами самов...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	ВАСИЛЕНКО ІРИНА ВІТАЛІЇВНА	УДК : 37.0 34:37.3:3	ВИХОВАННЯ ГРОМАДЯНСЬКОСТІ СТУД ЕНТІВ ЗАСОБАМИ САМО...	13.00.07
201002	22	Дарчик С.В.	Візія національно-визвольної війни українського на...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	ДАРЧИК Світлана Василівна	УДК 930.2 (477) "1	ВІЗІЯ НАЦІОНАЛЬНО-ВИЗВОЛЬНО Ї ВІЙНИ УКРАЇНСЬКОГО Н...	07.00.06
201002	25	Олексін М.І.	Вплив дисперсії і нелінійних ефектів оптичного вол...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	Олексін Михайло Іванович	УДК 621.391 .6 : 535	ВПЛИВ ДИСПЕРСІЇ І НЕЛІНІЙНИХ ЕФЕКТІВ ОПТИЧНОГО ВОЛ...	05.12.02
201002	29	Красівська В.В.	Вплив на гемостаз паталогічних інгібіторів зсіданн...	http://irbis-nbuv.gov.ua//cgi-bin/irbis64r_81/cgii...	СІВСЬКА Валерія Валеріївна	УДК : 616.15-06+616.	ВПЛИВ НА ГЕМОСТАЗ ПАТОЛОГІЧНИХ ІНГІБІТОРІВ ЗСІДАНН...	14.01.31
201002	30	Плахотнюк	Вплив стану молочної залози	http://irbis	ПЛАХОТНЮК ІГОР	УДК 619:618.11	ВПЛИВ СТАНУ МОЛОЧНОЇ	16.00.07

Рис. 8. Фрагмент результирующей таблицы извлечения данных

информацию, полученную в результате парсинга html-страницы (авторефераты за месяц-год) и текстов самих авторефератов, где:

yearmonth – значение года месяца поступления автореферата;

id – идентификатор автореферата;

author – фамилия и инициалы автора авторефераты, извлеченные из html-страницы запроса;

name – название автореферата (диссертации), извлеченное из html-страницы запроса;

link – адрес ссылки, где находится полный текст автореферата;

fio – фамилия имя и отчество. извлеченные из текста автореферата;

udc – значение классификатора УДК, извлеченное из текста автореферата;

nazvanie – название автореферата (диссертации), извлеченное из текста автореферата;

speciality – значение классификатора специальность, извлеченное из текста автореферата.

В дальнейшем нам нужно научиться извлекать из плоского текста персоны и организации. Можно использовать готовые библиотеки. Есть желание их сравнить и реализовать методы самостоятельно. При извлечении именованных сущностей можно использовать такие ресурсы (мы называем их Словарями), как Википедия и Викидата для верификации этих именованных сущностей. Поскольку, например, организации уже есть как в Википедии, так и в базе данных Викидата.

5. Хранение данных

Набор атрибутов персоналий не фиксирован и может неограниченно расширяться, поэтому необходима модель данных, учитывающая эту особенность. Поскольку конечной целью является представление данных в Интернете, желательна модель данных легко совместимая с семантической сетью. Подходящей моделью данных является RDF.

Выводы

В работе рассмотрены подходы к решению задачи сбора и извлечения разрозненных данных о персоналиях из слабоструктурированных и неструктурированных документов, представленных в общедоступных каталогах авторефератов диссертаций.

На языке PHP с применением XPath разработана система, которая позволяет автоматически собирать первичные документы из электронного каталога Национальной библиотеки Украины им. В.И. Вернадского, извлекать из этих документов данные и сохранять их в локальном хранилище.

Для хранения выбрана модель данных RDF с учётом особенностей данных и возможностью последующего представления в семантической сети.

Литература

1. Проскудина Г.Ю., Кудим К.А. О технологии использования внешних данных при создании и редактировании энциклопедических текстов. *Проблеми програмування.* 2017. № 1. С. 67–82.
2. Markus Krötzsch, Michael Günther, Markus Damm & Georg Wild (2016). SQID - Searching, Querying, and Interacting with Data. [online – https://tools.wmflabs.org/sqid].

References

1. PROSKUDINA G. & KUDIM K. (2017). About technologies of use of external data on creating and editing of encyclopedic text. *Problems in programming*. [online – pp.isofts.kiev.ua] (1). P. 67–82. (in Russian). Available from: http://pp.isofts.kiev.ua /ojs1/article/view/223 [Accessed 6/02/2017].
2. MARKUS KRÖTZSCH, MICHAEL GÜNTHER, MARKUS DAMM & GEORG WILD (2016). SQID - Searching, Querying, and Interacting with Data. [online – https://tools.wmflabs.org/sqid]. (in English). Available from: http://pp.isofts.kiev.ua/ojs1/ article/view/145 [Accessed 6/06/2017].

Получено 10.05.2019

Об авторах:

Кудим Кузьма Алексеевич,
младший научный сотрудник.
Количество научных публикаций в
украинских изданиях – 16.
Количество научных публикаций в
зарубежных изданиях – 2.
http://orcid.org/0000-0001-9483-5495,

Проскудина Галина Юрьевна,
научный сотрудник.
Количество научных публикаций в
украинских изданиях – 29.
Количество научных публикаций в
зарубежных изданиях – 15.
http://orcid.org/0000-0001-9094-1565.

Место работы авторов:

Институт программных систем
НАН Украины,
03187, Киев-187,
проспект Академика Глушкова, 40.
Тел.: (044) 526 3559.
E-mail: guproskudina@gmail.com,
kuzmaka@gmail.com

УДК 004.855:519.216 https://doi.org/10.15407/pp2019.02.047

О.С. Балабанов

АНАЛІТИКА ВЕЛИКИХ ДАНИХ: ПРИНЦИПИ, НАПРЯМКИ І ЗАДАЧІ (ОГЛЯД)

Висвітлено основні напрямки, задачі та типи результатів глибокого аналізу великих (комп'ютеризованих) даних. Показано практичне значення великих даних та великої аналітики як фундаменту створення нових комп'ютерних технологій планування і керування у бізнесі. Виділено специфічні для великих даних режими використання даних (або роди завдань аналізу): «інтелектуальний» пошук потрібної інформації; масована переробка («відпрацювання») даних; індукція моделі об'єкту (середовища); екстракція знань з даних (відкриття структур і закономірностей). Окреслено етапи і організацію циклу робіт з аналізу даних. До типових класів задач великої аналітики належать: групування випадків (кластеризація); виведення ціле-визначених моделей (класифікація, регресія, розпізнавання); виведення генеративних моделей; відкриття структур і закономірностей. Охарактеризовано особливості «глибокого навчання» та фактори його популярності. Виділено каузальні мережі як клас моделей, які поєднують у собі переваги генеративних, ціле-визначених та багатоцільових моделей і відрізняються тим, що придатні для прогнозу ефектів керування (втручання). Вказано шість «опор», на яких будується методологічне ядро великої аналітики.

Ключові слова: великі дані, аналіз даних, виведення моделі, відкриття знань, статистичні методи, предиктивні та генеративні моделі, каузальна мережа, прогноз.

Великі дані. Роль і значення

Великі дані (Big Data) стали помітним феноменом розвитку інформаційних технологій пост-індустріального суспільства і впливають на різні аспекти життєдіяльності, від політики до наукових досліджень [1–12]. Останні 10–15 років позначені безпрецедентним зростанням електронних (комп'ютеризованих) зібрань даних з різноманітних сфер діяльності. Це явище характеризують як повінь даних. Є багато варіантів визначення великих даних; не повторюючи їх, нагадаємо головні особливості, що спонукали ввести поняття великих даних. Великі дані (ВД), по-перше, відрізняються такими величезними обсягами, що їх зберігання, супроводження, управління і доступ до них наштовхується на обмеження існуючих технічних й програмних засобів [13]. Проблеми й перешкоди подекуди мають не тільки технічний, а й принциповий характер. Тож, робота з великими даними потребує нових нетрадиційних рішень. Обсяги накопичених сьогодні електронних даних вимірюються зетабайтами (тобто величинами порядку 10^{21} байт) [4–10]. Деякі дані генеруються (породжуються) з такою швидкістю, що маємо тільки дві альтернативи – або втрачати ці дані, або записувати їх негайно й такими, якими вони виміряні (сприйняті). Тому великі дані часто характеризуються як «швидкі», «сирі», неструктуровані й неточні. Можна сказати, що ВД виникли внаслідок впровадження автоматичних засобів та механізмів, які швидко й майже безупинно вимірюють та реєструють цифрові (електронні) дані з відповідного середовища (обладнання). Результати вимірів автоматично записуються «за годиться», хоча переважна маса даних залишиться не спожитою і згодом буде стерта. Назвемо типові середовища й джерела, звідки походять великі дані. На сьогодні відомі наступні джерела ВД: сенсорні мережі, прилади промислових об'єктів та технологічних ліній виробництва, торгові центри (супермаркети), інфраструктурні системи (енергетичні, транспортні тощо), соціальні мережі в Інтернеті, YouTube з «океаном» відеофайлів, системи on-line продаж, мобільний зв'язок, біржі та інші фінансові центри, навколоземні супутники спостережень, різноманітні датчики та прилади контролю навколишнього середовища та екологічних служб, прилади відео-спостереження, файли зображень з автоматизованих телескопів, устаткування експериментальних досліджень з фізики частинок, прилади біомедичних обстежень (зокрема, МРТ-зобра-

ження), дані біохімічних вимірів (генетика, протеоміка) і т. д. Побіч того, великі дані породжуються в результаті переведення у цифрову форму даних державних, адміністративних та суспільних реєстрів, медичних карток, статистичної звітності й т. д.

Актуальність аналітики великих даних визначається прискоренням збору і накопиченням великих масивів емпіричних даних з різноманітних сфер діяльності суспільства, а також готовністю наукових, програмних й комп'ютерних ресурсів для створення аналітичних продуктів. У розвинутих країнах світу в цих дослідженнях й розробках задіяні величезні ресурси й численні наукові та інженерні кадри [5–11]. Комплекс досліджень та розробок під назвою «великі дані плюс велика аналітика» не є абсолютно новим явищем. Його можна сприймати як продовження (або новий етап) того алгомеративного й інтегративного напрямку розвитку методів, засобів й технологій, який називали Data Mining, Knowledge Discovery in Data, інтелектуальний аналіз даних, виділення знань з даних і т. п. Багато положень, методів та напрацювань, отриманих «під дахом» названих понять, залишаються адекватними й корисними для ВД [14–18]. Водночас існує низка особливостей, що характеризують новизну ВД й великої аналітики. Якщо два десятиліття тому доступ та підготовку даних розглядали як допоміжний етап, то тепер пошук, доставка й попередня обробка великих даних стають все більш проблемним етапом усього циклу використання даних. Пріоритет зусиль зміщується на інструменти й технології пошуку, доступу до потрібних «сирих» даних та підготовки релевантних даних (маніпуляції з ВД).

Діапазон впроваджень великої аналітики охоплює бізнес, державне управління та наукові дослідження. Підтверджується теза про зсув у методології досліджень. Підвищується роль індуктивно-емпіричного підходу. Можна казати, що формується парадигма прискореного пізнання на основі узагальнення емпіричних даних [6, 14, 15, 19–22]. Доступність всебічних релевантних даних дозволяє автоматизувати процес наукового відкриття, і деякі автори проголосили настання четвертої ери в історії науки [21]. Традиційне публічне наукове обговорення залишається необхідним, але його акцент зміщується з етапу висунення гіпотез і процесу вироблення положень і рішень на етап оцінки й інтерпретації результатів (теорії) та їх інтеграції в систему наукових знань.

Витрати на збір і зберігання даних величезного обсягу виправдані тільки якщо ті дані будуть результативно використані і забезпечать достатнє відшкодування (зиск). Деякі потреби можна задовольнити окремими записами, вилученими з великих даних. Мається на увазі, що кінцевим результатом стають відповідні файли, фрагменти чи записи, відібрані з масиву даних в тій формі, в якій вони зберігаються. Наприклад, для розслідування злочину потрібні окремі записи в журналах або кадри відео-спостереження. Проте іноді знайти у ВД потрібну інформацію стандартними засобами (зокрема, через SQL-запити) важко. Справа не тільки у обсягах вмістилища даних і проблемах доступу до них. Часто неможливо точно описати, що саме аналітик (користувач) хоче знайти, важко сформулювати запит. Але головний напрямок використання ВД – іншого характеру. Основний шлях результативного використання ВД здійснюється через глибокий аналіз даних, коли величезний масив сирої інформації перетворюється («перетравлюється»), і на виході видається компактна, концентрована й цінна інформація кінцевого споживання. З даних вилучається (екстрагується) їх цінний сенс. Отже, великі дані «автоматично» передбачають велику аналітику (ВеАн).

Часто організація (фірма, орган управління) має у своєму розпорядженні великі зібрання даних, але ці дані дуже вибірково та обмежено залучаються до процесу досліджень, підготовки планів чи прогнозування наслідків пропонованих управлінських рішень. Вибір і обґрунтування рішень традиційно робилися на основі експертних суджень і оцінок, адекватність й актуальність яких важко контролювати. (Крім того, часто експертні міркування та суб'єктивні уявлення покладають

в основу побудови математичних моделей). Зазвичай адекватна модель є невідома, а знання про об'єкт існують як сукупність розрізнених відомостей та уявлень вузьких спеціалістів. Таку «скирту інформації» важко узгодити та звести у робочу модель.

Доступність ВД дозволяє отримати широкий спектр інформації про об'єкт та середовище. З'являється можливість побудувати замкнений комп'ютеризований цикл планування і керування. Вихід ВД на ринок ІТ-продуктів дозволяє кардинально оновити технологію й практику підготовки й обґрунтування важливих рішень. Нова технологія рішень будується як комп'ютерна, з визначальною роллю даних (data-driven), що дозволяє позбутися консерватизму й суб'єктивізму у керуванні. Рішення для керівництва фірми виробляються як прямий результат аналізу й переробки комплексу різноманітних релевантних даних (наприклад, про процеси продаж, поведінку споживачів, про діяльність підрозділів фірми тощо).

Підготовка планів, прогнозування наслідків рішень і дій, а також інші аналітичні дослідження мають безпосередньо ґрунтуватися на аналізі масивів емпіричних даних. Актуальна задача – ідентифікація потрібної адекватної моделі «об'єктивними» методами на основі зібраних даних спостережень. Шукана модель приречена бути емпіричною (за витоками) та феноменологічною (за змістом і формою подання).

Комп'ютеризація цілого циклу менеджменту має вирішальне значення для маркетингу популярних й стрімко обновлюваних продуктів (гаджетів, засобів побутового комфорту тощо). Сотні тисяч компаній скористалися ресурсами, сервісами та аналітичними засобами Amazon Web Services, побудованими на хмарних технологіях зберігання даних та обчислень. Перелічимо деякі приклади застосувань ВД, згадані в статтях [5–10, 12, 23–26].

В корпорації Шеврон проаналізували терабайти сейсмічних даних мексиканської затоки, поліпшили свої комп'ютерні моделі, і в результаті підвищили успішність буріння від 1 з 5-ти спроб до 1 з 3-х. (Одне буріння вартує 100 мільйонів доларів). Деякі страхові компанії тепер не тільки відстежують добробут й майно клієнтів, а й також збирають дані сенсорів, вмонтованих у автомобілі, й аналізують кілометраж, маршрути, час поїздок тощо. Транспортна компанія U.S. Xpress підтримує моніторинг сенсорних даних про стан та місцезнаходження своїх авто й вантажівок, а також дані з мобільників та гаджетів водіїв та операторів. Дані накопичуються у хмарі й аналізуються. За результатами аналізу оптимізується керування усім автопарком. Вантажівки вчасно спрямовуються до ближчих заправок пального з нижчою ціною. Для техобслуговування авто-засоби спрямовуються до оптимально підібраних депо. Враховуються затори на дорогах, потреба розігріву мотору взимку і т. д. Кілька інших фірм й агентств аналізують великі дані для оптимізації логістики та постачання енергії.

Служби й агентства з охорони здоров'я інтегрують дані з різних джерел. Медичні зводи доповнюються й уточнюються даними індивідуального рівня з соціальних мереж, даними викликів медслужб по мобільним телефонам тощо. Інтеграція клінічних даних з даними поведінки та суспільними показниками допомагає знизити вартість та підвищити якість лікування. Для оцінки заходів з охорони здоров'я у провінції Квебек дослідники зіставили медичні записи з даними продаж продуктів харчування в тому ж регіоні (спираючись на поштовий код). Покупки з використанням карток лояльності дозволили прив'язати споживачів до місць проживання. Зіставлення адміністративних даних з індивідуальними даними (релевантними до стану здоров'я) дозволило уточнити статистику захворювань діабетом. Аналіз соціальних мереж та пошукових слів в Інтернеті дозволив значно підвищити оперативність моніторингу розповсюдження легеневих інфекцій. Відомий факт, що аналіз «твітів» допоміг простежити розповсюдження холери.

Провайдери мобільного сервісу аналізують демографічні дані споживачів,

статус їх житла, деталі користування сервісами, що дозволяє надавати оптимальні персоналізовані пропозиції телекомунікаційного сервісу. Також провайдери розробляють систему оперативного виявлення обману (використовують предиктивну аналітику). Менеджери центрів роздрібної торгівлі тепер аналізують не тільки кошики покупок, але й потоки покупців з розбивкою на соціальні групи. Компанії роздрібної торгівлі on-line (зокрема, Amazon) для персоналізованих рекомендацій аналізують пошукові слова користувача, його «кліки» протягом сеансу, покупки у минулому тощо. Великий список посилань на застосування великих даних можна знайти в [1, 5].

Процес організації великої аналітики (обрис)

Цінність великих «сирих» даних визначається нашою здатністю вилучати з них «сенс», корисний за змістом і зручний за формою. Практика вимагає виділяти цінний екстракт швидко, використовуючи «свіжі» дані. Коли сукупність доступних даних охоплює екстремальне широкий спектр інформації, фірма (організація) може виконувати багато оперативних функцій автоматизовано, майже повністю на основі ВД. Отже, треба будувати замкнений комп'ютеризований цикл технологій – від збору даних до кінцевого застосування результатів (рішень, керування). «Непрозорі» й не-комп'ютерні процедури виносяться за межі «оперативного» циклу керування. (За штабами фірми залишаються функції нагляду (супервізія) та вищий рівень керування.) Виконання аналітичного завдання завершується видачею моделі або результату в формі, придатній для кінцевого застосування. (Вживають термін «actionable outputs».) Такий результат може використовуватися протягом певного періоду, коли виконується «короткий» цикл аналітики (для керування використовують «свіжі» дані звуженої номенклатури). Схема циклів життя ВД та ВеАн (великий цикл, цикл аналізу, цикл використання) зображена на рис. 1.

Оскільки ВеАн використовує переважно статистичні методи, дані мають складатися з списку випадків (прикладів), що характеризують однотипні об'єкти або той самий об'єкт у варіабельних умовах. Випадки можуть трактуватися як екземпляри популяції, прецеденти, транзакції, цикли та періоди функціонування. (Існують дані, де поняття випадків та прикладів не збігаються [16]). Більшість традиційних методів аналізу потребують, щоб дані всіх випадків складалися з єдиного набору ат-

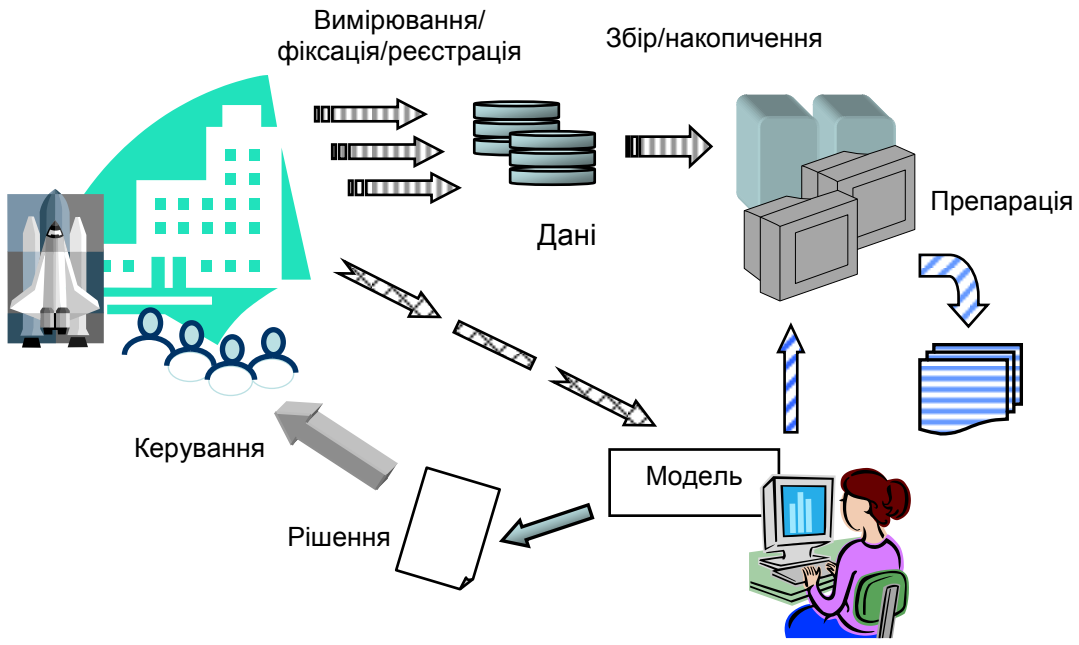


Рис. 1. Цикли великої аналітики і використання даних

рибутів і збиралися за єдиною схемою вимірювання. Більшість класичних методів й процедур аналізу даних розраховані на зручно форматовані дані (зазвичай – у формі таблиці), що вміщуються в пам'яті комп'ютера. Натомість ВД наповнені переважно «сирими», різнорідними, неузгодженими, невпорядкованими та неструктурованими даними. Інформація щодо певного випадку може знаходитися у різних файлах і сховищах. Іноді доводиться розглядати як «випадок» не тільки вектор чисел, а й цілий образ, текст, структуру і т. д. В деяких даних неясно, як розрізнити і виділити окремі випадки.

Дані, що зберігаються у сховищах, можна поділити на: 1) структуровані; 2) «гнучко-структуровані» або слабко-структуровані; 3) неструктуровані. До структурованих відносять дані, організовані за жорсткою схемою. Кожна одиниця (запис) даних складається з уніфікованого набору позицій, і кожну позицію займає елемент (атрибут) відповідного відомого змісту. (Часто це елементи одного типу, наприклад, дійсні числа.) Така структуризація гарантує прості й «прозорі» процедури імпорту даних в усіх платформах. Гнучко-структурованими можна назвати дані, де не зафіксовано набору позицій для елементів. До цього виду належать дані широкого спектру, включаючи довільні послідовності символів, графові структури, мовні тексти й гіпертексти. До гнучко-структурованих треба віднести також дані, які побудовані за рекурсивними схемами (з невизначеними розмірами). Текст має свою структуру, визначену синтаксисом, граматикою та іншими обмеженнями, але така структуризація не забезпечує однозначної інтерпретації елементів (слів) і не підтримується стандартними процедурами обробки. Неструктуровані дані не мають чітко визначеної структури. Для використання неструктурованих даних потрібні нестандартні процедури конверсії, спеціальна розмітка, додаткові дескриптори і т. п. Схожі проблеми виникають, коли дані структуровані, але структура фіксації даних нерегулярна і невідповідна (або невідома аналітику). Маємо проблеми, коли не тільки фізична, але й логічна структура даних не збігається із змістовною («семантичною») структурою. Такі дані виникають, наприклад, коли записується потік сигналів або коли об'ємне зображення описується простою послідовністю точок (пікселів). Можна виділити також дані з частково-невідомою структурою. Деякі дані можна інтерпретувати та «зрозуміти» тільки з допомогою «автора» даних. Неструктурованість та різнорідність даних створює певні проблеми для обробки [6–10, 16, 27, 28]. Потрібні попередні етапи компіляції та інтеграції даних.

Доволі частою є ситуація, коли окремі прилади (засоби) автономно збирають дані про ті самі індивіди популяції чи про ті самі (або еквівалентні) транзакції, цикли функціонування об'єкту, а зібрані дані накопичуються в окремих файлах. Для того, щоб могли працювати типові методи аналізу даних, потрібно співвіднести (ідентифікувати) відповідні записи в різних файлах і сформувати «випадки». Але це не вдається зробити, якщо в файлах немає інформації, яка допомогла би однозначно розпізнати і ототожнити прецеденти (випадки). З точки зору багатовимірного аналізу, маємо ситуацію вертикально-секціонованих, («розщеплених») даних. Отже, з метою отримання з ВД корисного «сенсу» перед власне результативним аналізом необхідно виконати відбір та підготовку даних [7].

Процес великої аналітики включає два етапи:

1) доставка та компіляція даних (пошук, добір, фільтрація, агрегація, комплектування, інтеграція, зменшення розмірності, синхронізація, переформатування);

2) власне глибокий аналіз підготовлених даних.

Ланцюг проходження завдання BeAн показано на рис. 2. Етап глибокого аналізу даних у свою чергу може складатися з ланцюга завдань. Попередня обробка може залучати методи, які традиційно розглядалися як методи власне аналізу даних (аналіз головних компонент, random projection і т. д.).

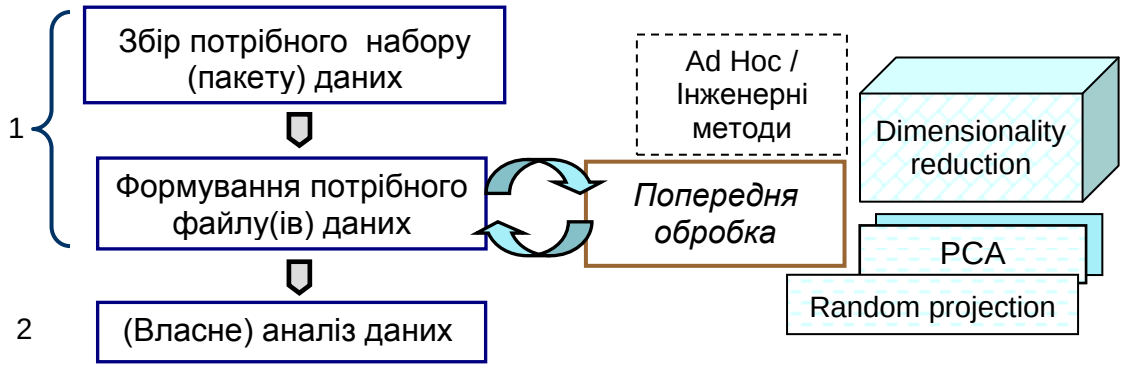


Рис. 2. Схема процесу великої аналітики

Одна з тенденцій BeAн – перенесення аналітичних засобів в програмне забезпечення баз даних, аби виконувати значну частину роботи в місцях зберігання, без передачі даних на сервер аналітика. Зокрема, фірма SAS у співпраці з Oracle та Teradata інтегрує свою аналітики в програмне забезпечення баз даних. Виконання аналітики прямо на платформі баз даних дає можливість розосередити, розподілити виконання задачі й використати паралелізм. (Такий режим може бути вимушеним у зв'язку з захистом даних.) Але такий режим далеко не завжди прийнятний з огляду на розмаїття методів аналізу, інструментарію й мов програмування. До того ж, масовану ітеративну переробку (з багаторазовим скануванням активної порції даних) зазвичай ефективніше виконувати, маючи ректифікований файл в локальній пам'яті комп'ютера аналітика. В деяких платформах та інструментах застосовується режим in-Memory Analytics, коли «гарячі» дані утримуються в пам'яті RAM (не переміщуються на диск).

Існує інструментарій, який інтегрує бази даних підрозділів (департаментів) у єдину систему даних фірми (організації) [16, 17, 26]. Одна з передових сучасних програмних платформ аналізу ВД (яка підтримує увесь цикл аналізу) – Apache Hadoop and MapReduce. Багатий комплект методів і програм аналізу даних для вказаної платформи надається відкритим середовищем Apache Mahout та Apache Spark [7–10, 26, 29]. Популярне відкрите середовище з мовою **R** (наступниця мови **S**) інтенсивно поповнюється програмами розв'язання різноманітних задач. В [30] наводиться низка пакетів програм та застосувань аналітики у біомедичній галузі. В роботі [5] надано перелік фірм-постачальників платформ для ВД.

Перспективні режими та інтелектуалізація аналізу даних

Великі дані надають ресурси і створюють передумови для виходу за межі можливостей стандартних методів пошуку інформації (SQL-запити, ключові слова). На черзі розробка технологій «інтелектуального» пошуку інформації (ІнтлПІ). Результат виконання ІнтлПІ може виглядати традиційно, тобто як запис, фрагмент чи цілий файл даних в тій формі, як він зберігається в базі (звісно, видається композиція знайдених фрагментів даних). Але суттєва відмінність полягає в тому, що користувач може скористатися інтелектуальним пошуком, коли не знає, як сформулювати запит (хоча він знає, що йому потрібне). Йдеться про ситуацію, коли інформаційну потребу не вдається конкретизувати через атрибути та характеристики реальних баз даних. Користувачу недостатньо навіть мета-даних, дескрипторів даних та онтологій для того, щоб специфікувати завдання. Релевантні атрибути та характеристики даних будуть ідентифіковані в процесі виконання ІнтлПІ, тільки після аналізу великого зрізу даних і виявлення певних відношень між багатьма фрагментами даних. Чи є запис даних релевантним залежить не тільки від вмісту цього запису, а й від вмісту інших записів і файлів. Наведемо фік-

тивний приклад гіпотетичного завдання для ІнтлПІ.

«Виявити в якійсь з розвинутих країн сукупність трьох наступних явищ. 1) Невмотивоване різке згортання виробництва сучасного виду озброєння або злам тренду фінансування таких озброєнь (останніми роками). 2) Незадовго перед тим в тій самій країні – суттєві кадрові перестановки в центральному департаменті озброєнь. 3) Одночасно – раптове припинення потоку публікацій з одного з перспективних напрямків науково-технічних досліджень.»

Замість «озброєння» в такому «запиті» могло би фігурувати інше високотехнологічне обладнання. Виконати подібний «запит» автоматично, без участі аналітика, мабуть, неможливо. Спроба звести таке завдання до послідовності SQL-запитів (навіть якби всі дані зберігалися у реляційних базах) була би безперспективною. Намагання відразу формалізувати подібний «запит» напевно призведе до квазі-логічної конструкції з кількома невідомими («вільними» змінними) і нечіткими поняттями. Щоби розпочати виконання запиту, необхідна діалогова взаємодія аналітика з системою. Аналітик має замінити природно-мовний запит пакетом стандартних завдань (де залишаться «вільні» змінні). Як попередню і автономну гілку пошуку можна запустити перегляд преси з метою знайти «відголоски», «сигнали» шуканого явища. («Сигнали» можуть утворюватися як набори таких слів, як «бюджет», «відставка», «полігон», «випробування», «перерозподіл ринку» і т. д.).

Інтелектуальний пошук інформації відрізняється від «інтенсивного» чи «розширеного» режимів традиційного пошуку. В ході «інтенсивного» пошуку аналіз виконується автономно («замкнено») в межах кожного запису чи файлу. (Приклади – пошук особи за фотороботом у великій базі зображень; пошук через Google.) «Розширення» режиму пошуку досягається надбудовою засобів, які враховують розподілену та агломеративну структуру зберігання даних та підтримують прості процедури поповнення даних (наприклад, NoSQL, HDFS та інші) [1, 5, 8, 10, 29]).

Взагалі, можна виділити наступні роди завдань з повномасштабним використанням ВД:

1) розширені режими традиційного пошуку інформації;

2) «інтелектуальний» пошук потрібної інформації (скомпонованих фактів, записів, фрагментів файлів);

3) масована проміжна переробка даних (чи краще сказати – «відпрацювання») однотипною процедурою за один-два проходи через масу даних (mining, concentration, – аналогія із збагаченням руди);

4) індукція моделі об'єкту (джерела), звідки взято дані;

5) екстракція знань з даних (відкриття структур і закономірностей).

«Відпрацювання» даних може бути призначене для підготовки даних перед наступним етапом екстракції знань. Прикладом власне «проміжної» переробки є обчислення достатніх статистик. Альтернативно, якщо на вході задати достатньо інформативні апріорні знання, то в режимі «відпрацювання» можна виробляти кінцевий результат («проміжна» переробка обертається на кінцеву). Далі в огляді в основному розглядаються завдання типу індукції моделей та екстракції знань.

Для проведення повномасштабних емпірико-індуктивних досліджень, які відштовхувалися б від «сирих» даних і доводили результати до рівня «кристалізованих» знань, необхідно побудувати багаторівневу високоорганізовану інтегровану технологію, з адекватними мовами спілкування між рівнями. Така технологія демонструватиме властивості, які вважаються інтелектуальними [20], і зможе підміняти (а в чомусь й перевершувати) людину-аналітика.

Повномасштабний процес виділення знань з даних дозволяє в одному великому (можливо – ітеративному) циклі аналізу здійснити те, що раніше (за посередництва аналітика) виконувалося набором завдань розвідкового (експлоративного) та конфірмативного аналізу даних.

Основні напрямки і задачі великої аналітики

Останніми роками на ринок засобів підтримки бізнесу виходять IT-продукти та інструментарій, що кардинально оновлюють технологію менеджменту та вироблення рішень, ґрунтуючи їх на аналізі ВД. Зрозуміло, найбільше враження справляють такі інструменти й технології, де вихідним результатом аналізу даних є практичний висновок, кінцева рекомендація чи навіть варіант бізнес-рішення. Образно кажучи, найбільш привабливою є переробка даних за схемою «стимули–рефлекс» (подібно до того, як регулюється поведінка тварини або елементарні акти поведінки людини). Тобто на виході технології отримуємо вказівку до дії («actionable output»). Така схема (суцільна «чорна скриня») працює для спеціальних задач, наприклад, розпізнавання (де ціль вказана, а ролі змінних в принципі відомі). Але стосовно проблем управління такий рівень «самостійності» та «самодостатності» інформаційної технологій може бути практично ефективним лише для дуже елементарного рівня «миттєвого» управління або за ідеалізованих умов (наприклад, за надзвичайно високої спеціалізації діяльності фірми у дуже стабільних умовах ринку). Для більшості практичних ситуацій така схема роботи нереалістична. Було б контрпродуктивно намагатися занурити у «чорну скриню» увесь процес вироблення рішень організації, цикл керування складним об'єктом або ціле дослідження. Більш реалістичне й корисне завдання для комп'ютера – знайти в даних цікаву (для користувача) інформацію, виділити закономірності, відкрити знання, побудувати «портрет» об'єкту у середовищі, вивести модель, яка відтворює систему зв'язків та впливів (показує, «як все розгортається»). А вже на основі отриманих «знахідок» (знань) аналітик і користувач зможуть виробляти вказівки до дії.

Задача глибокого аналізу даних визначається метою аналітика (користувача) і типом потрібного результату. Це має вказати аналітик (програміст) [14, 15, 20, 22, 27]. Іноді строго сформулювати завдання і дати постановку задачі важко. (Навіть для таких конкретних задач, як розпізнавання об'єктів чи образів, часто не формулюють строгої постановки.) Тому в багатьох випадках доречно вести мову не про постановку задачі, а про проблемну ситуацію. Коли аналітик остаточно не визначився з постановкою, можна виконувати розвідкові або стандартні завдання (з «підручного меню»).

Традиційно дані є багатовимірною статистичною вибіркою і подаються у формі плаского масиву. Масив даних має «ширину» та довжину («вишину»). В ширину розташовано набір змінних (атрибутів) X. Мабуть, найбільш «загальне» завдання – стисло описати дані в форматі X. Зрозуміло, що аналітика цікавить не буквальний опис даних $\mathbf{X}$, а опис системи змінних, очищений від гамору і випадкових домішків. Отже, типове завдання (з «підручного меню») – вивести модель даних у формі сумісного розподілення ймовірностей $p(X)$. (Цю задачу часто називають unsupervised learning [31].) Результат такого типу вважається «генеративною моделлю» даних в слабкому сенсі (про генеративну модель в сильному сенсі буде далі). Декларувати таке завдання просто, але коли маємо змінні дійсного типу, і тих змінних багато, і не задано параметричної форми для $p(X)$, тоді незрозуміло, як описувати $p(X)$. Крім того, сама по собі модель в формі $p(X)$, як правило, не цікава. Така модель буде цінною й цікавою, тільки якщо розподілення $p(X)$ демонструє яскраві особливості (часткове виродження, специфічну форму, нестандартні ознаки, неочікувані паттерни), і ці особливості можна компактно описати і змістовно (предметно) інтерпретувати. Коли немає чіткої мети, краще не намагатися знайти повний опис $p(X)$, а дати спрощений опис, характеристику $p(X)$ «в загальних рисах». Наприклад, обчислити моменти, парні коваріації і т. п. (подібні характеристики можна обчислити за одне сканування даних). Така задача – сумарізації даних. Для темпоральних даних корисна інформація – основна частота коливань. Задачі сумарізації даних перетинаються із задачами зниження вимірності даних. Іноді ко-

рисно замість прямого опису даних знайти «прообраз» цих даних, виражений через гіпотетичні змінні Z. (Опис даних $p(X)$ можна відтворити майже без втрат через стандартне перетворення «прообразу» $p(Z)$.) Репрезентація через «прообраз» зацікавить аналітика, якщо гіпотетичні змінні Z взаємонезалежні або їх кількість помітно менша за кількість оригінальних змінних X. Компактну репрезентацію даних може надати класична задача аналізу головних компонент (principal component analysis – PCA) [31]. Нелінійний аналог PCA – виділення принципових кривих, принципових поверхонь тощо. Ще один варіант спрощеного не акцентованого аналізу – шукати тільки інтервали максимальних значень ймовірності $p(X)$. Спеціальним випадком такої задачі можна вважати пошук правил асоціації (у разі дискретних змінних) або узагальнених асоціацій. (Втім, така задача постала як суто прикладна (market basket analysis), без загальної постановки. Виведення правил асоціації можна розглядати як задачу виявлення паттернів.)

Масив даних $\mathbf{X}$ не завжди є статистичною вибіркою за схемою i.i.d. Наявні дані можуть походити з різних популяцій (з різних «моделей»). Тоді важливіше виділити компоненти суміші, а не виводити єдину генеративну модель для суміші. Іноді виділити компоненти можна за допомогою кластеризації (теж завдання з підручного меню). В загальному випадку розділити суміш проблематично (функції щільності ймовірностей компонент можуть значною мірою перетинатися.)

Шанси знайти цікаві, корисні, а головне – практично потрібні результати (моделі, регулярності, паттерни) значно зростають, якщо завдання вдало специфіковане і на вході задано адекватну апріорну інформацію. Традиційні методи аналізу даних були розраховані на вибірки даних малого та середнього розміру. В таких ситуаціях отримати корисний результат можна тільки за умови, що «в загальних рисах» модель задана апріорі. Взагалі, для того, щоб отримати на виході переробки даних змістовний та обґрунтований результат, необхідно подати на вхід сукупність знань та емпіричних даних, таку, що вони «в сумі» утворюють достатньо багату інформацію про об'єкт. Різні співвідношення вказаних двох складових вхідної інформації породжують різні проблемні (когнітивні) ситуації, а відтак – і різні роди задач аналізу даних. Для спрощення задачі традиційно задавали обмеження на вході. Що менше на вході апріорних знань й обмежень, то більшим має бути вміст й обсяг даних.

Додаткова інформативність результату відносно вхідної апріорної інформації («додана вартість» на виході) завжди менше, ніж інформація, що міститься в просканованих та оброблених даних. Часто ця засвоєна інформація становить лише мізерну частку інформації, яка пройшла через процесор, внаслідок того, що дані значною мірою не релевантні для задачі (щодо мети), а також через те, що цінний вміст даних занадто захаращений гамором невідомого характеру. Зростання обсягів даних дозволяє розширити можливості виведення моделей з даних, вивести більш точну й адекватну модель, а також обійтися без жорстких обмежень та важких чи ризикованих припущень на вході.

Мабуть, найпростіший і найпоширеніший спосіб визначити мету й акцент завдання – вказати цільову змінну (характеристику, атрибут) y. Зазвичай змінна y присутня в даних, але іноді її приписує аналітик перед виконанням завдання (робить «розмітку»). В задачах класифікації та розпізнавання цільова змінна y дискретна, а в задачах типу регресії – неперервна. Такі задачі (їх іноді називають supervised learning) можна назвати цілеспрямованими або ціле-визначеними («націленими»). Ціле-визначена задача виводить результат (модель) у формі $p(y|X)$ або $y=\Phi(X)$. (Строго кажучи, оскільки маємо $y\in X$, то треба писати $y=\Phi(Z)$, де $Z\subseteq X\setminus\{y\}$.)

Опис $\Phi(\cdot)$ не обов'язково (не завжди) є аналітичною чи явно вираженою функцією. Це може бути алгоритм, процедура чи просто «чорна скриня». Коли цільова змінна y дискретна, модель вигляду $y=\Phi(X)$ називають «дискримінативною», на противагу «генеративній» моделі $p(X)$.

Результат у формі $p(y|X)$ або $y=\Phi(X)$ також часто називають «предиктивна» (predictive, «передбачувальна») модель. Відповідно, (суто формально) кажуть про предиктивну аналітику. Втім, така модель не обов'язково призначена для «передбачень» майбутніх подій. Зазвичай «предикція» спрямована радше назад у часі (наприклад, при класифікації чи розпізнаванні). Тоді краще сказати не предикція, а відтворення значення змінної y. В науковій літературі предиктивними моделями (та методами) називають такі, які мають здатність до узагальнення, тобто претендують на збереження адекватності поза обробленими даними (на відстані від врахованих прикладів). Іншими словами, до «дійсно предиктивних» моделей ставиться вимога, щоб вони забезпечували адекватну екстраполяцію у просторі прикладів (попри те, що ці моделі виведені із скінченої вибірки даних, яка має вибірковий ухил). В спільноті прагматиків побутує ще більш звужене й вимогливе розуміння предиктивних моделей та «предиктивної аналітики». А саме, характеристика «предиктивна» вживається до методів побудови моделей, наближених до практики бізнесу [24, 32]. В цій спільноті предиктивна аналітика розуміється як така, що забезпечує умовне прогнозування наслідків управління об'єктами, передбачення подій та майбутньої поведінки реальних складних динамічних систем, що розвиваються. До речі, виведення результатів типу «actionable outputs» подекуди називають «прескриптивна» аналітика. А до дескриптивної аналітики відноситься сумарізація даних.

Результат у формі $y=\Phi(X)$ передбачає включення в опис тільки необхідних (значущих) факторів (предикторів, ознак, аргументів, коваріат, регресорів). Отже, коли ставиться ціле-визначена задача, майже завжди мається на увазі, що треба виконати відбір значущих змінних серед заданого набору. Відтак, предиктивні (дискримінативні) моделі схильні до меншої розмірності, ніж «генеративні». Більш того, розв'язання багатьох прикладних задач (класифікації, розпізнавання) передбачає формування ознак, тобто із заданих на вході змінних шляхом комбінації й інтеграції формують нові змінні (підвищеного рівня), які входять в кінцеву «модель» $y=\Phi(X)$.

Останніми роками спостерігається справжній бум досліджень і розробок методів так званого «глибокого навчання». В цих задачах задається не тільки цільова змінна y, але й схема моделі (див. далі). Глибокому навчанню можна протиставити глибокий аналіз даних та відкриття знань. До глибокого аналізу даних відносимо групи задач, які мають на меті:

- відтворити «портрет» об'єкту у середовищі, тобто вивести модель, яка «прозоро» інтерпретується і пояснює функціонування об'єкту;
- відкрити структуру в даних, наприклад, ідентифікувати систему зв'язків та впливів між характеристиками об'єкту у середовищі;
- знайти закономірності поведінки системи (об'єкту) – регулярність, періодичність, інваріанти; знайти аномалії.

Коли завдання спрямоване на відкриття знань, аналітик зазвичай не вказує цільову змінну. Але навіть якщо аналітик задав цільову змінну y, його метою не обов'язково є побудова предиктивної (дискримінативної) моделі $y=\Phi(X)$ чи прогнозування значень змінної y для певних випадків (умов). Метою може бути ідентифікація факторів (причин), які об'єктивно визначають значення y. Задачі відкриття знань та виведення моделей з емпіричних даних (з мінімальними припущеннями на вході щодо майбутнього результату) можна назвати індуктивною переробкою даних. Уявлення про індукцію статистичної моделі як відтворення адекватного опису процесу генерації даних започаткував ще Р. Фішер. Адекватний опис процесу генерації даних – це «портрет» джерела даних (а відтак – і об'єкту). Він допомагає зрозуміти, що саме й чому відбувається.

Відмінність між виведенням моделі об'єкту та відкриттям знань в даних можна характеризувати наступним чином. По-перше, задача відкриття знань ставиться з суттєво меншою вагою апріорних знань на вході (принаймні щодо предмету відкрит-

тя). По-друге, модель, як правило, претендує на опис всіх оброблених даних (із застереженням, що коли йдеться про предиктивну модель, вона включає не всі представлені, а тільки релевантні змінні.) Натомість «знання» (результат процесу екстракції знань) є паттернами, які не завжди підтримуються всіма даними, але повторюються достатньо регулярно. (З точки зору статистики, це такі паттерни, що частота їх підтвердження в даних суттєво перевищує рівень, який можна було би пояснити випадковістю.) Закономірність може стосуватися лише окремого зрізу (чи сегменту) даних, проте виконується систематично. Третя відмінність «знань» – вони мають пізнавальну імпресивність. Знання й паттерни відображають яскраві особливості, які легко інтерпретуються і є цікавими в пізнавальному сенсі. Натомість модель у формі $y = \Phi(X)$ може не показувати нічого цікавого, але вона продуктивно «працює».

В результаті глибокого аналізу даних можуть бути знайдені імплікативні правила вигляду: (вектор характеристик $\mathbf{A}$) => (окрема_характеристика_B). Такий результат теж вкладається у форму $b = \Phi(A)$, хоча може залучати не аналітичні, а логічні вирази. Такий результат вважається знахідкою й знанням тільки в тому разі, якщо правило дає значення b з високою точністю (детерміновано), і до того ж змінні b та A були автоматично знайдені, а не задані на вході. Аналогічну ситуацію маємо щодо кластеризації: про відкриття знань доречно говорити тільки якщо знайдені кластери є статистично значущі. Базовий принцип виділення регулярностей – знайдення часто повторюваних сполучень, паттернів, схем, або навпаки, занадто рідких сполучень («дірок» у розподіленні). Критерієм регулярної повторюваності (або регулярної відсутності) є значне відхилення від статистично очікуваних значень (або від очікувань аналітика). Аби отримані результати можна було прийняти як адекватні знання (закономірності) про об'єкт, необхідно запобігти сценарію, коли ті результати (неявно) закладено в процедурах виведення, або коли вони є артефактами збору чи попередньої обробки даних.

На рис. 3 запропоновано один з варіантів систематизації великої аналітики за родами задач та типами результатів.

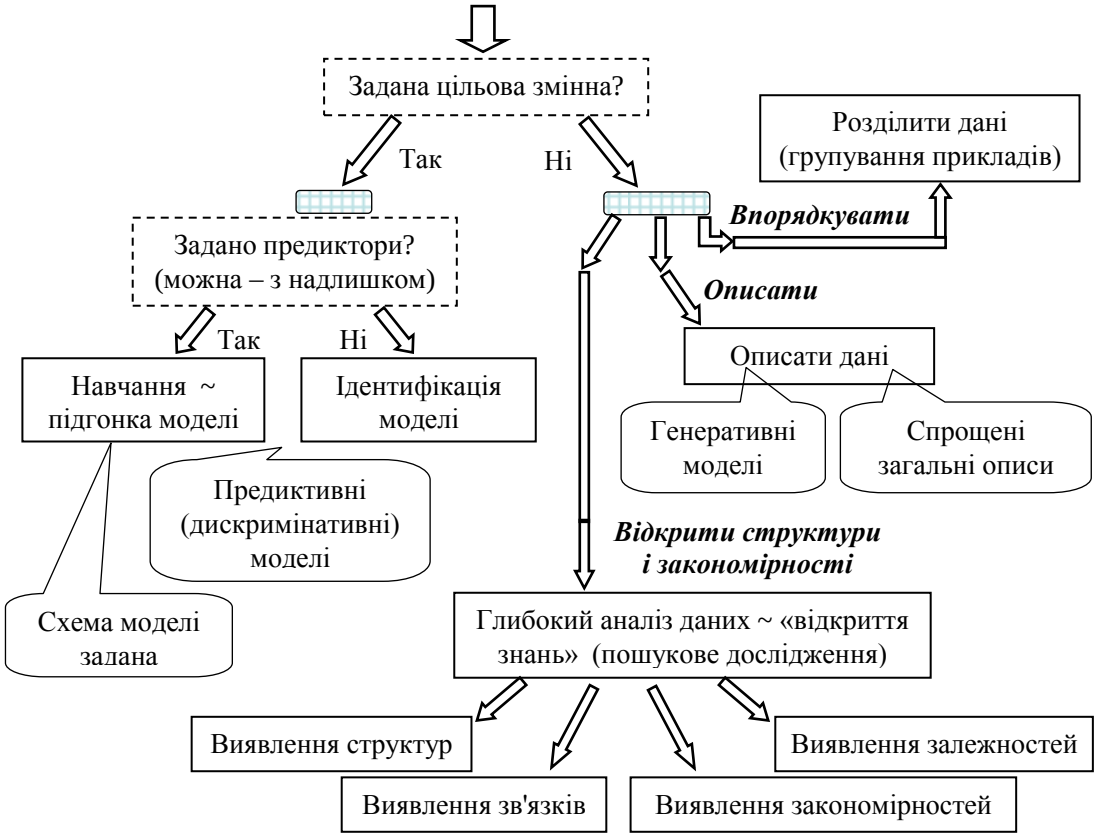


Рис. 3. Типи задач та результатів великої аналітики

Типові задачі аналізу даних

Вибірково окреслимо підходи до аналізу слабко-структурованих даних. Методи аналізу текстів екстрагують потрібну інформацію з новин, оглядів, електронних листів, «твітів», документів, статей. Текст – це не емпіричні дані; текст не задовольняє припущень, прийнятих в традиційних методах аналізу даних. (Втім, великі зібрання документів можуть розглядатися як вибірка екземплярів популяції й аналізуватися статистично.) Залишимо поза увагою комп'ютерну лінгвістику та технології, спеціалізовані на мовах. Прості методи кількісного аналізу тексту виловлюють лише «поверхову» інформацію. Внутрішня структурованість тексту незручна для традиційних методів аналізу. Відомо дві групи простих методів аналізу текстів: 1) екстракція інформації; 2) сумарізація текстів [7]. Методи першої групи розпізнають у тексті об'єкти (сутності) та виділяють відношення між ними. Методи сумарізації текстів застосовують дві техніки. «Екстрактивна сумарізація» робить компіляцію вирізок (фрагментів) заданого тесту, враховуючи місце та частоту входження слів. «Абстрактивна сумарізація» намагається виявити семантику тексту і може видати результат в інших термінах і конструкціях. Щоб автоматично з'ясувати зміст і сенс тексту, залучаються методи обробки природної мови. Розробляються також методи генерування природомовних відповідей на запитання, а також методи розпізнавання опіній та настроїв, які приховані «між рядками» тесту. Методи аналізу даних з Web-середовища можна знайти у [33]. Введення в аналіз даних соціальних медіа дається в [34]. В роботі [35] описано, зокрема, аналіз даних в інформаційних мережах, збагачених текстом. Огляд аналізу даних з Інтернету речей можна знайти в [35, 36].

В аналітичних задачах та у побудові моделей широко вживаються поняття зв'язку і відношення. Але в різних контекстах зв'язок має дуже відмінний сенс. Перелічимо відомі тлумачення поняття зв'язку, які зустрічаються в літературі з комп'ютерних наук та інформаційних технологій. Отже, типи зв'язку: логічна залежність (зчеплення окремих значень); статистична залежність (зчеплення частот значень); суміжність; близькість; посилання («лінки», адресація); (безпосереднє) слідування у часі; відношення «об'єкт – атрибут (ознаки)»; «ціле – частина (деталь)»; відношення приналежності (до класу). Залежність має семантичні градації: асоціація, вплив, каузальний зв'язок.

Можна запропонувати наступний перелік типових задач ВеАн:

1) групування випадків (записів, об'єктів); кластеризація;

2) виведення ціле-визначених моделей (для класифікації, регресії, розпізнавання);

3) виявлення регулярних паттернів (систематичних повторювань):

– структурних, зокрема, послідовних (motifs), 3-вимірних, графових,..

– наборів (правил асоціацій, item sets, market baskets), .. ;

4) виявлення типових (для популяції) дискретних послідовностей у часі (лінки, ланцюги дій тощо);

5) виявлення трендів, періодичності та аномалій (в даних із темпоральною прив'язкою);

6) відтворення структур залежностей;

7) відтворення каузальних моделей.

Впорядковані у часі дані (ряди даних) не є статистичною i.i.d.-вибіркою у буквальному розумінні (хоча за певної трансформації теж можуть розглядатися як стандартна вибірка). Темпоральні дані (в першу чергу для неперервних процесів) надають простір для специфічних задач аналізу, наприклад, виявлення періодичності, трендів, динамічних аномалій [37]. Знайдені тренди та періодичність у даних допомагають виконувати «феноменологічний» безумовний (інерційний) прогноз. Інші регулярні паттерни також допомагають прогнозувати у відповідних ситуаціях. Натомість знання каузальної моделі дає аналітичний інструмент для прогнозування наслідків втручання в об'єкт (керування). Моделі розпізнавання або класи-

фікації (в першу чергу ті, що побудовані як нейронні мережі) радше надають не знання, а вміння. До виявлення знань можна зарахувати хіба що підбір підмножини значущих предикторів. Але більшість традиційних методів розв'язують цю задачу у дуже спрощеному й спеціальному варіанті.

Вузька спеціалізація традиційних ціле-визначених моделей випливає не тільки з фіксації цільової змінної, але й з прив'язки до формату кандидатів у предиктори (фактори). Нехай для заданих y та X була виведена модель $\hat{y} = \Phi(Z)$, де $Z \subseteq X$. Можлива ситуація, коли потрібно оцінити (спрогнозувати) значення y за умови, що відомі значення тільки деяких факторів, тобто змінних Q, причому $Q \subset Z$. Як застосувати модель в цій ситуації? Якщо $\Phi(\cdot)$ – формула, що підставити в формулу на місце невідомих факторів? Якщо $\Phi(\cdot)$ – процедура, що подати на її відповідні входи? Простої задовільної відповіді на ці питання немає. Треба враховувати кореляцію між факторами, а також їх взаємодію всередині моделі. Поставлене питання знаходить коректну відповідь в апараті каузальних мереж, який дозволяє адаптувати модель до будь-якого формату запиту. (Звісно, вказана проблема неактуальна для тих задач розпізнавання, де вхідні дані характеризуються великою надлишковістю та дублюванням. Наприклад, втрата якихось точок (пікселів) зображення компенсується на етапі вироблення ознак за рахунок сусідніх точок.)

Каузальні моделі

Для забезпечення адаптивності моделі до формату запиту потрібно знати адекватну картину зв'язків між всіма задіяними змінними. (Це потрібно також для ідентифікації справжніх причини для заданого ефекту.) Для задач планування та управління потрібна модель, яка допомагає зрозуміти зв'язки та взаємозалежності між окремими субпроцесами у реальному середовищі об'єкту. Бажано, аби виведена модель була придатна для прогнозу наслідків виконання рішень менеджера (керування). Вказаним вимогам відповідають каузальні моделі і, зокрема, каузальні мережі [38–41]. Факторний аналіз та аналіз незалежних компонент (ICA) знаходять структуру як сукупність незалежних прихованих змінних, які спільно (адитивно) формують значення наявних змінних. Натомість каузальні мережі описують структуру безпосередніх впливів між наявними змінними (зазвичай – в умовах неповної спостережуваності). В процесі виведення каузальної моделі з'ясовується (розпізнається) каузальний характер статистичних зв'язків (кореляцій, асоціацій, залежностей) [39, 42–44]. Стислу характеристику властивостей каузальних мереж можна знайти в [19, 38, 44, 45]. Одним з варіантів КМ є кореляційна мережа для фінансової аналітики [46].

Каузальна мережа (КМ) – це модель залежностей між змінними, яка адекватно відображає структуру спрямованих впливів. КМ описується як пара (G, Θ), де G – граф, що специфікує структуру моделі, Θ – параметри, прив'язані до G, які описують кількісний аспект моделі. В практичних задачах використовують структури без орієнтованих циклів (тобто орграф G – ациклонний). Обмежимося класом моделей з одно-орієнтованими ребрами, тобто на основі ординарних ациклонних орграфів (оАОГ). Множина параметрів оАОГ-моделі складається із сукупності локальних параметрів, заданих для кожної змінної. Зокрема, в мережі, що показана на рис. 4, для змінної Y опис може виглядати як $y = \mathrm{f}(x, z, v) + \varepsilon_Y$. Функція $\mathrm{f}(\cdot)$ може мати будь-яку форму, але зручніше мати справу з лінійною залежністю (що автоматично означає адитивність моделі й індивідуальну прив'язку коефіцієнтів до ребер моделі).

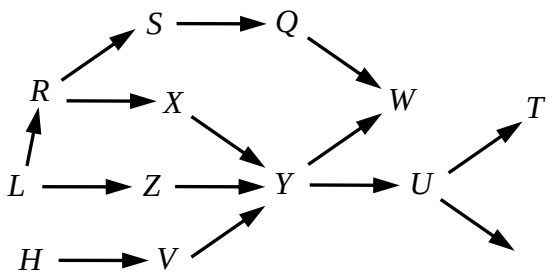


Рис. 4. Приклад каузальної мережі

Каузальні мережі поєднують у собі переваги моделей кількох типів. За умови адекватності, КМ є генеративними моделями в сильному сенсі, – вони адекватно описують процес генерації змінних, ізоморфно («дзеркально») відображаючи процеси в об'єкті. КМ також є предиктивними і дискримінативними моделями, тому що застосовують описи у формі $p(y|X)$ або $y = \mathrm{F}(X)$. Більш того, кожна КМ є багатоцільовою моделлю, оскільки вміщує в собі сукупність ціле-визначених моделей (потенційно – для всіх форматів запиту). КМ можна назвати системою регресійних/класифікаційних моделей, інтегрованих «без швів» за допомогою відношень умовної незалежності.

Каузальні моделі допомагають висвітлити принципову відмінність впливу та асоціації, уточнюють роль та інформативність змінних. Розрізняються два режими використання моделі – «пасивна» предикція та каузальний прогноз («активна предикція»). «Пасивна» предикції розуміється як обчислення значення цільової змінної C, виходячи з значень асоційованих (пов'язаних) з нею змінних $A,B,\ldots$. Така задача формулюється як $p(C|a,b,..)$. Це звичний режим застосування традиційних моделей, зокрема, класифікації. Для пасивної предикції безумовно інформативними виступають всі суміжні змінні, безвідносно до характеру зв'язків (і причини, і наслідки). Наприклад, для моделі, що показана на рис. 4, для предикції (оцінки) значення Y інформативними є X, Z, V, U, W. Але корисний внесок в пасивний прогноз Y можуть зробити й несуміжні змінні, за умови присутності (відсутності) інших змінних у переліку заданих. Умовно- інформативними для Y є всі змінні, поєднані з Y якимось шляхом. Зокрема, якщо не задано значення змінних X, Z, то інформативними стають L та S. Що стосується змінної Q, то вона стає інформативною, якщо задано значення змінної W. (Звісно, внесок «далеких» змінних – незначний). Для пасивного прогнозу корисні не тільки справжні причини, а й тісно пов'язані з ними індикатори. Для класифікації часто використовуються змінні, які радше є наслідками («дітьми») або «братами» цільової змінної.

Каузальний прогноз відповідає на питання, яким буде значення заданої змінної, якщо маніпулювати (керувати) певними іншими змінними (точніше, їх прототипами в об'єкті). Для моделі, показаної на рис. 4, керування змінними Q,W,T не дасть ефекту для Y за жодних умов. Каузальний прогноз значення Y після втручання на змінну X потребує «усунення» внеску конфаудера L. Це здійснюється як корекція моделі (видалення зв'язку $R \to X$). Взагалі, каузальний прогноз для C за втручання на змінну A формулюється як $p(C|do(a),b,..)$ [38–41, 43]. Отже, каузальні мережі є предиктивними моделями у сильному сенсі.

Завважимо, що КМ утворюється зі змінних, заданих на вході, тож і прогноз виражається через них. Але на основі результатів, отриманих з моделі, можна обчислювати «кінцевий» (з точки зору замовника) ефект, для чого залучаються додаткові («зовнішні») функції і фактори, що залишилася поза вхідними даними. (Методи відтворення каузальних мереж з даних згодом будуть розглянуті детальніше.)

Самонавчання алгоритмів та глибоке навчання

У спеціальній літературі часто вживається термін Machine Learning, який зазвичай перекладають буквально – машинне навчання. Під гаслом Machine Learning велися розробки алгоритмів, процедур, методів і програмних засобів розв'язання практичних задач протягом майже пів століття [47]. Ці розробки зосереджувалися на ціле-визначених задачах (оцінка успішності навчання потребує задану ціль). Напрямок Machine Learning (ML) окреслився після того, як дослідники й інженери зрозуміли, що для багатьох прикладних задач (зокрема, класифікації) важко придумати (вибрати) ефективний алгоритм розв'язання. З'ясувалося, що замість того, щоб «вручну» специфікувати потрібний алгоритм розв'язання, краще вирішити задачу вищого рівня – задачу адаптивного конструювання потрібного алгоритму самим

комп'ютером. Тобто запускається автоматичний процес конструювання «цільового» алгоритму як послідовність вибору опцій та параметрів в ході пробних застосувань алгоритму розв'язання кінцевої задачі на «прикладах». Приклади задаються вхідними даними. Підбір опцій диктується успішністю розв'язання прикладної задачі, а весь цей процес називається навчанням. Отже, предметом того, що позначають терміном Machine Learning, є способи і методи автоматичного формування («навчання») алгоритмів і засобів розв'язання прикладних задач на основі досвіду їх розв'язання на прикладах. Коротко це можна назвати «самонавчання алгоритмів» (сНАлг).

Словосполучення Machine Learning широко розповсюдилося в літературі. Щодо вживання «машинне навчання» як терміну можна зауважити наступне. По-перше, воно може дезорієнтувати, бо таке словосполучення стосується також застосування комп'ютерів у навчальному процесі. По-друге, вживання слова «машина» тут не є влучним.

Результатом виконання сНАлг зазвичай є алгоритм обчислення y на основі X (хоча іноді може бути видана модель у певній декларативній формі). Напрямок сНАлг сприймався як такий, що входив до комп'ютерних наук (методів програмування і обчислень) і часто позиціонувався «під дахом» напрямку «штучний інтелект». (До речі, в розвитку самого штучного інтелекту пріоритет змістився від «вилучення» знань (тобто отримання їх від експерта) до виведення знань з даних.) В руслі робіт з сНАлг було винайдено багато способів, тактик, правил, методик й методів, переважно інженерно-евристичних [16, 17, 47–49]. Зокрема, розвинуто інструментарій нейронних мереж. Часто розробники обходилися без математичної постановки задачі, і тільки останнім часом почали запозичувати зі статистики принципи та підходи для обґрунтування, оцінки статистичної значущості та оцінки надійності.

В руслі напрямку нейромереж сформувалася гілка методів так званого «глибокого навчання». Методи глибокого навчання застосовуються переважно для візуального та звукового розпізнавання [50]. В цьому підході опріч цільової змінної y (якою зазвичай є клас об'єкту або «сигнальна» характеристика розпізнавання), за замовчуванням задається й інші апріорна інформація. Вхідні характеристики є кандидатами у предиктори (ознаки) або радше їх компонентами (елементами). Задано також форми перетворення (параметричні родини моделей) або арсенал «цеглин» (будівельних блоків), з яких можна конструювати «модель» $y = \Phi(X)$. Часто задано параметри конструкції «моделі» (кількість рівнів, блоків). Список кандидатів у предиктори (фактори) може бути надлишковим, але зазвичай всі кандидати однакові зі рівнем деталізації і мають однаковий «статус» (це зрозуміло за «фізичним» змістом). Висока спеціалізація «моделі» дозволяє добре специфікувати завдання. По-суті, для глибокого навчання задано «каркас» моделі. «Глибина» в цьому підході означає ієрархічність, багаторівневість конструкції, а також складність використаних функцій (формул). Характер даних диктує необхідність спочатку сформувати з вхідних змінних більш інформативні масштабні ознаки, на основі яких вже побудувати модель. Глибоке навчання продемонструвало, що на відповідному класі задач можна натренувати багаторівневі конструкції, які адекватні при застосуванні до нових прикладів (об'єктів). Успішність глибокого навчання пояснюється характером проблемної ситуації, а саме, наступними обставинами. Вхідні змінні – це дуже «дрібні ознаки» (маленькі частинки «картини», наприклад, пікселі зображень). Велика кількість змінних, причому «сусідні» змінні тісно корельовані і майже ідентичні. Модель високоспеціалізована, з лаконічним результатом на виході (одне з кількох значень). На вході задано «каркас» моделі.

У глибокому навчанні «узагальнення» має сенс об'єднання деталей у ціле, а у глибокому аналізі – радше перехід від одиничного до загального. У глибокому навчанні глибина розуміється як багаторівневість і складність конструкції. Натомість у глибокому аналізі

даних глибина розуміється як сходження від «сирих» випадкових даних до «знань» (до очищеної зрозумілої «картини»), причому ті «знання» випливають з відносин між змінними, характер і роль яких невідомі (змінні можуть бути дуже різнорідними). Ці два напрями різняться також характером переробки даних: перший – це тренування, «підгонка» й оптимізація; другий має пошуково-дослідницький характер [16].

Огляд основних методів аналізу та особливостей їх застосування до великих даних буде презентовано у наступній статті.

Велика аналітика. Проміжні підсумки

Великі дані є одним зі знакових трендів новітніх інформаційних технологій у розвинутих країнах. Великі дані породжуються швидкісними автоматичними засобами реєстрації інформації, вбудованими в реальні об'єкти. Витрати на збір та зберігання великих даних виправдовуються їх результативним використанням, в першу чергу – через глибокий аналіз даних, коли величезний масив сирих даних перетворюється («перетравлюється») на компактну, концентровану й цінну інформацію кінцевого споживання. Аналіз може бути глибоким тільки коли є багата і рясна «сировина».

Взагалі, великі дані можуть бути використані у наступних режимах: «інтелектуальний» пошук інформації; масована переробка даних («відпрацювання», concentration, mining) за один-два проходи; виведення моделі об'єкту (джерела) з даних; екстракція знань з даних (відкриття закономірностей).

Деякі фірми вже впроваджують замкнені комп'ютеризовані технології, що охоплюються увесь цикл оперативного керування – від збору даних до кінцевого застосування (рішень). Великі дані є родючою сировиною для глибокого аналізу (принаймні для аналізу зв'язків) тільки коли вони багатовимірні. Великі дані в принципі можуть забезпечити інформацію, достатню для планування і знайдення оптимальних рішень. Проте потенційна «повнота» даних часто залишається «віртуальною». Великі дані часто є неструктурованими, «гнучко-структурованими» або слабко-структурованими. Крім того, великі дані часто є вертикально-секціонованими («розщепленими»). Перед власне результативним аналізом необхідно виконати підготовку даних. Цей етап може включати такі процедури, як пошук, добір, доставка, фільтрація, агрегація, інтеграція, синхронізація, пере-форматування. Водночас іноді потрібно зменшувати вимірність даних (без втрати їх змістовності).

Можна очікувати, що у майбутньому технології збору даних прогресують, пристрої стануть «тямущими», а інфраструктура розростеться у масштабах. Це забезпечить постачання багатовимірних інтегрованих даних, готових для негайного аналізу. Проте проникнення таких засобів у життя суспільства буде входити у суперечність з правом на приватність й конфіденційність.

Велика аналітика увібрала багатий арсенал кількох дисциплін та набутки різних напрямків розробок. Вона спирається на фундамент статистичної методології (включаючи розвідковий та конфірматив-ний аналіз даних), методи оптимізації та пошуку, методи репрезентації знань та візуалізації багатовимірних даних. Адаптується досвід таких напрямків, як відкриття знань в даних (Data Mining, Knowledge Discovery in Data) і методи самонавчання алгоритмів (Machine Learning). Кілька напрямків досліджень і розробок стали опорами і складовими великої аналітики (рис. 5). Їх об'єднання і взаємне збагачення утворює методологічне ядро великої аналітики.

Типові класи задач аналітики включають: розділення даних (групування випадків); поверховий («загальний») не акцентований опис даних; виведення цілевизначених моделей; відкриття структур та закономірностей. Ціле-визначені задачі охоплюють виведення предиктивних (дискримінативних) моделей, які описують цільову змінну через інші змінні.

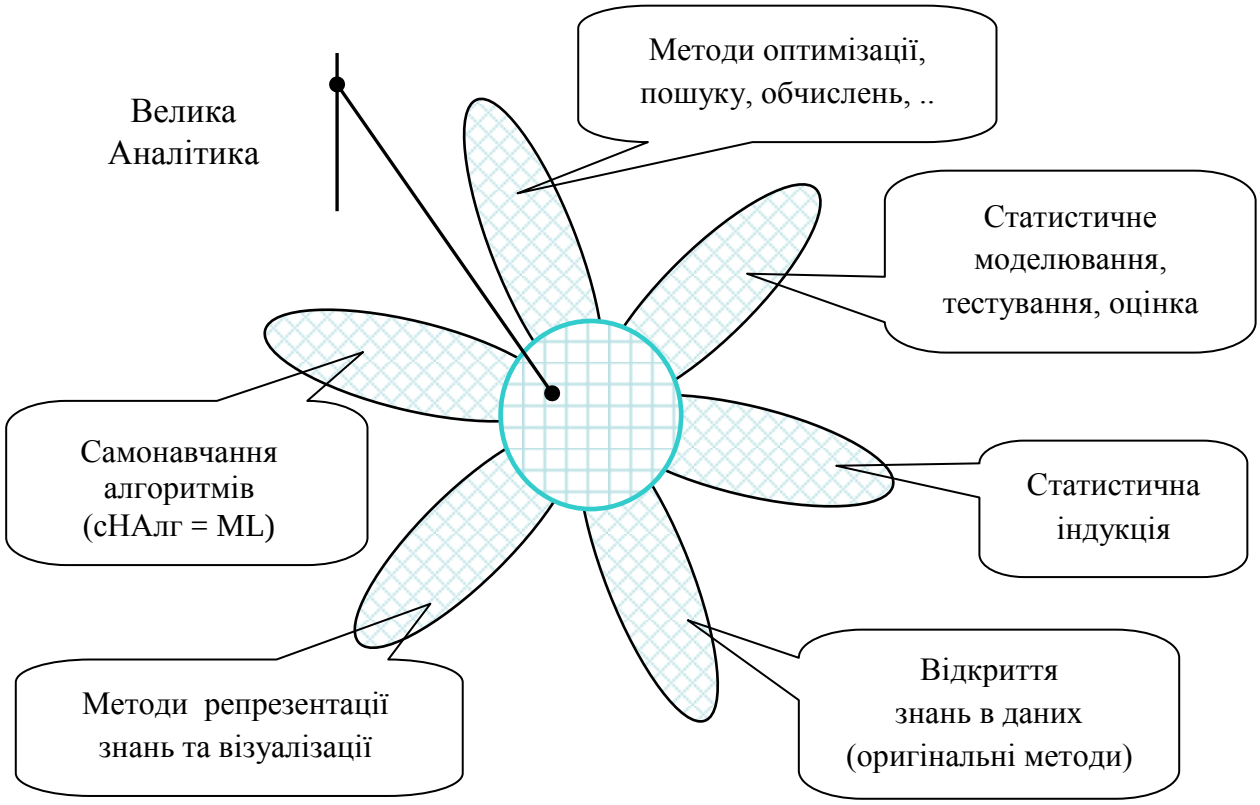


Рис. 5. Фундамент та арсенал великої аналітики

Відмінність моделі об'єкту (результату виведення) і «знання» (результату відкриття) характеризується в трьох аспектах. 1) На вході задачі відкриття знань задається менше апріорної не емпіричної інформації. 2) Модель, як правило, претендує на опис всіх оброблених даних, в той час як «знання» може стосуватися лише окремого зрізу (чи сегменту) даних (проте підтверджується систематично). 3) «Знання» змістовно інтерпретуються, цікаве, неочікуване або надає «інсайт», водночас як модель виконує передбачену функцію або описує дані.

Критичним питанням для адекватності ціле-визначених моделей є підбір значущих предикторів. Модель з високою предиктивною ефективністю не завжди дає розуміння (пояснення) предмету. Популярним різновидом ціле-визначених задач є так зване «глибоке навчання», призначене для розпізнавання образів та мови. Успішність «глибокого навчання» пояснюється спеціальним характером задачі розпізнавання та вхідних даних. Глибокому навчанню можна протиставити глибокий аналіз даних та відкриття знань. У «глибокому навчанні» глибина розуміється як багаторівневість і складність конструкції, а у глибокому аналізі даних – як сходження від «сирих» випадкових даних до «знань», причому ті «знання» не є артефактами алгоритмів виводу чи збору даних, а є результатом «кристалізації» зв'язків, розчинених в масі даних. Форми виявлених закономірностей включають: послідовнісні повторювання (motifs), періодичність коливань індикаторів у часі, інваріанти на основі комбінації характеристик, часто повторювані набори (асоціації), структури залежностей тощо.

Каузальні мережі є генеративними моделями в сильному сенсі, бо вони здатні адекватно описати процес генерації змінних, «дзеркально» відображаючи процеси в об'єкті. Каузальні моделі пристосовані для застосування в режимі варіювання набору заданих значень предикторів (умов). Головна перевага каузальних моделей над традиційними – вони підтримують прогнозування наслідків втручання в об'єкт (керування).

Великі дані надають нові можливості для статистичних методів аналізу і вод-

ночас висувають вимоги до них [16, 23, 24, 28, 30, 31, 32, 51–60]. Результати аналізу великих даних потребують оцінки й верифікації за статистичними принципами. Розповсюдження великих даних стимулює подальший розвиток методів аналізу (зокрема, статистичних) та прогрес комп'ютерних технологій.

Література

1. Big data analytics: a survey. Tsai C.-W., Lai C.-F., Chao H.-C. and Vasilakos A.V. *Journal of Big Data*. 2015. Vol. 2, N. 1. P. 1–32.
2. Science in the petabyte era. *Nature* (journal). 2008. Vol. 455, Issue 7209. Springer Nature Ltd.
3. Frankel F., Reid R. Big data: Distilling meaning from data. *Nature*. Vol. 455, September 2008. p. 30.
4. Doctorow C. Big data: Welcome to the petacentre. Ibid. P. 16–21.
5. Chen C.L.P. and Zhang C.-Y. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*. 2014. Vol. 275. P. 314–347.
6. Cukier K. Data, data everywhere: A special report on managing information. *The Economist*. 2010, February 25.
7. Gandomi A. and Haider M. Beyond the hype: Big data concepts, methods, and analytics. *Intern. Jour. of Information Management*. 2015, Vol. 35, N. 2. P. 137–144.
8. Watson H.J. Tutorial: Big Data analytics: Concepts, technologies, and applications. *Comm. of the Association for Information Systems*. 2014. Vol. 34, Article 65. P. 1247–1268.
9. Sivarajah U., Kamal M.M., Irani Z. and Weerakkody V. Critical analysis of Big Data challenges and analytical methods. *Journal of Business Research*. 2017. Vol. 70. P. 263–286.
10. Bhadani A. and Jothimani D. Big Data: Challenges, opportunities and realities / In.: M.K. Singh and D.G. Kumar (eds.). Effective Big Data management and opportunities for implementation. IGI Global, USA, 2016.
11. Intern. Journal of Data Science and Analytics. Special issue on Data Science in Europe. 2018. Vol. 6, Issue 3. P. 163–269.
12. Intern. J. of Data Science and Analytics. Spec. issue on environmental and geospatial data analytics. 2018. Vol. 5, Issue 2–3. P. 81–211.
13. Jacobs A. The pathologies of big data. *Comm. of the ACM*. 2009, Vol. 52, Issue 8, P. 36–44.
14. Андон Ф.И., Балабанов А.С. Выявление знаний и изыскания в базах данных: подходы, модели, методы и системы (обзор). *Проблемы программирования*. 2000, № 1–2. С. 513–526.
15. Балабанов А.С. Выделение знаний из баз данных – передовые компьютерные технологии интеллектуального анализа данных. *Математичні машини і системи*. 2001, № 1–2. С. 40–54.
16. Data mining: practical machine learning tools and techniques / I.H. Witten, F. Eibe, M.A. Hall. (3rd ed.). Morgan Kaufmann, San Francisco, CA. 2011. 629 p.
17. Data Mining. A Knowledge Discovery Approach. K.J. Cios, W. Pedrycz, R.W. Swiniarski and L.A. Kurgan. Springer, 2007, 606 p.
18. Azzalini A. and Scarpa B. Data analysis and Data Mining: An introduction. Oxford University Press, N.Y., 2012. 288 p.
19. Андон Ф.И., Балабанов А.С. Структурные статистические модели: инструмент познания и моделирования. *Системні дослідження та інформаційні технології*. 2007, № 1. С. 79–98.
20. Балабанов О.С. Комп'ютерний інтелект: фантастичні перспективи і щоденний поступ. 1997, revised 2007. [Електронний ресурс.] Доступ: https://www.researchgate.net/publication/332269445_KOMP'UTERNIJ_INTELEKT_FANTASTICNI_PERSPEKTIVI_I_SODENNIJ_POSTUP
21. Hey T, Tansley S. and Tolle K. The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research, Redmont, WA. October 2009. 252 p.
22. Siebes A. Data science as a language: challenges for computer science — a position paper. *Intern. J. of Data Science and Analytics*. 2018. Vol. 6. P. 177–187.
23. Fan J., Han F. and Liu H. Challenges of Big Data analysis. *Nat. Scient. Rev*. 2014. Vol. 1, N. 2. P. 293–314.
24. Statistical inference, learning and models in Big Data / B. Franke, J.-F. Plante, R. Roscher, E.A. Lee, C. Smyth, A. Hatefi, F. Chen, E. Gil, A.G. Schwing, A. Selvitella, M.M. Hoffman, R. Grosse, D. Hendricks and N. Reid. *Intern. Statistical Review*. 2016. Vol. 84, N 3. P. 371–389.

25. Swanson N.R. and Xiong W. Big Data analytics in economics: What have we learned so far, and where should we go from here? *Canadian Journal of Economics*. 2018. Vol. 51, Issue 3. P. 695–746.
26. The anatomy of big data computing / R. Kune, P. K. Konugurthi, A. Agarwal, R.R. Chillarige and R. Buyya. *Software: Practice and Experience*. 2016, Vol. 46. P. 79–105.
27. Smirnova E., Ivanescu A., Bai J., Crainiceanu C.M. A practical guide to big data. *Statistics and Probability Letters*. 2018. Vol. 136. P. 25–29.
28. Shi J.Q. How do statisticians analyse big data — our story. *Statistics and Probability Letters*. 2018. Vol. 136. P. 130–133.
29. Jiang H., Chen Y., Qiao Z., Weng T. H. and Li K.C. Scaling up MapReduce-based big data processing on multi-GPU systems. *Cluster Computing*. 2015. Vol. 18, N. 1. P. 369–383.
30. Haughton D. Software packages for data mining. *Wiley StatsRef: Statistics Reference Online*. 2016. P. 1–5.
31. James G., Witten D., Hastie T. and Tibshirani R. An introduction to statistical learning with applications in R. Springer, N.Y., 2013. 426 p.
32. Graham E. and Timmermann A. Forecasting in Economics and Finance. *Annual Review of Economics*. 2016. Vol. 8. P. 81–110.
33. Liu B. Web data mining: Exploring hyperlinks, contents, and usage data. Springer-Verlag: Berlin-Heidelberg, 2011. 622 p.
34. Zafarani R., Abbasi M.A. and Liu H. Social media mining. An introduction. Cambridge University Press. 2019. 380 p.
35. Big Data Analysis: New Algorithms for a New Society. N. Japkowicz and J. Stefanowski (eds.), Springer, Switzerland. 2016. 329 p.
36. Data mining for the Internet of things: Literature review and challenges. F. Chen, P. Deng, J. Wan, D. Zhang. *Intern. Journal of Distributed Sensor Networks*. Vol. 2015. 14 p.
37. Esling P. and Agón C. Time-series data mining. *ACM Computing Surveys*. 2012. Vol. 45, Issue 1. P. 12–34.
38. Pearl J. Causality: models, reasoning, and inference. Cambridge: Cambridge Univ. Press. 2000. 526 p.
39. Spirtes P., Glymour C. and Scheines R. Causation, prediction and search. New York: MIT Press, 2001. 543 p.
40. Балабанов О.С. Відкриття знань в даних та каузальні моделі в аналітичних інформаційних технологіях. *Проблеми програмування*. 2017, № 3. С. 96−112.
41. Peters J., Janzing D. and Schölkopf B. Elements of Causal Inference. Foundations and Learning Algorithms. MIT Press, Cambridge, MA, USA, 2017. 265 p.
42. Shiffrin R.M. Drawing causal inference from Big Data. *Proc. Nat. Acad. Scien. USA*. 2016. Vol. 113, N. 27. P. 7308–7309.
43. Pearl J. and Bareinboim E. External validity: From do-calculus to transportability across populations. *Statistical Science*. 2014. Vol. 29, N 4. P. 579–595.
44. Балабанов О.С. Від коваріацій до каузальності. Відкриття структур залежностей в даних. *Системні дослідження та інформаційні технології*. 2011, № 4. С. 104–118.
45. Балабанов О.С. Відтворення каузальних мереж на основі аналізу марковських властивостей. *Математичні машини та системи*. 2016, № 1. С.16–26.
46. Giudici P. Financial data science. *Statistics and Probability Letters*. 2018. Vol. 136. P. 160–164.
47. Machine learning. Special issue on applications of machine learning and the knowledge discovery process. R. Kohavi, F. Provost. (Eds.) *Machine Learning*. 1998. Vol. 30, N.2/3. P. 127–274.
48. 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, August 13–17, 2016. San Francisco, California.
49. 24th SIGKDD Conference on Knowledge Discovery and Data Mining, August 19–23, 2018. London, UK.
50. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. Vol. 521. P. 436–444.
51. Donoho D.L. 50 Years of Data Science. *Journal of Computational and Graphical Statistics*. 2017. Vol. 26, Issue 4. P. 745–766.
52. Bühlmann P. and van de Geer S. Statistics for high-dimensional data: Methods, theory and applications. Springer, 2011. 556 p.
53. Bühlmann P. and van de Geer S. Statistics for big data: A perspective. *Statistics and Probability Letters*. 2018. Vol. 136. P. 37–41.
54. Secchi P. On the role of statistics in the era of big data: A call for a debate. Ibid. P. 10–14.
55. Quarteroni A. The role of statistics in the era of big data: A computational scientist' perspective. Ibid. P. 63–67.
56. Cox D.R., Kartsonaki C., Keogh R.H. Big data: Some statistical issues. Ibid. P. 111−115.
57. James G. M. Statistics within business in the era of big data. Ibid. P. 155–159.

58. Weihs C. and Ickstadt K. Data Science: the impact of statistics. *Intern. Journal of Data Science and Analytics*. 2018. Vol. 6. P. 189–194.
59. Efron B. and Hastie T. Computer age statistical inference. Cambridge University Press, N.Y., 2016. 475 p.
60. Carmichael I. and Marron J.S. Data science vs. statistics: two cultures? *Japanese Journal of Statistics and Data Science*. 2018. Vol. 1, Issue 1. P. 117–138.

References

1. Big data analytics: a survey. Tsai C.-W., Lai C.-F., Chao H.-C. and Vasilakos A.V. *Journal of Big Data*. 2015. Vol. 2, N. 1. P. 1–32.
2. Science in the petabyte era. *Nature* (journal). 2008. Vol. 455, Issue 7209. Springer Nature Ltd.
3. Frankel F., Reid R. Big data: Distilling meaning from data. *Nature*. Vol. 455, September 2008. p. 30.
4. Doctorow C. Big data: Welcome to the petacentre. Ibid. P. 16–21.
5. Chen C.L.P. and Zhang C.-Y. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*. 2014. Vol. 275. P. 314–347.
6. Cukier K. Data, data everywhere: A special report on managing information. *The Economist*. 2010, February 25.
7. Gandomi A. and Haider M. Beyond the hype: Big data concepts, methods, and analytics. *Intern. Jour. of Information Management*. 2015, Vol. 35, N. 2. P. 137–144.
8. Watson H.J. Tutorial: Big Data analytics: Concepts, technologies, and applications. *Comm. of the Association for Information Systems*. 2014. Vol. 34, Article 65. P. 1247–1268.
9. Sivarajah U., Kamal M.M., Irani Z. and Weerakkody V. Critical analysis of Big Data challenges and analytical methods. *Journal of Business Research*. 2017. Vol. 70. P. 263–286.
10. Bhadani A. and Jothimani D. Big Data: Challenges, opportunities and realities / In.: M.K. Singh and D.G. Kumar (eds.). Effective Big Data management and opportunities for implementation. IGI Global, USA, 2016.
11. Intern. Journal of Data Science and Analytics. Special issue on Data Science in Europe. 2018. Vol. 6, Issue 3. P. 163–269.
12. Intern. J. of Data Science and Analytics. Spec. issue on environmental and geospatial data analytics. 2018. Vol. 5, Issue 2–3. P. 81–211.
13. Jacobs A. The pathologies of big data. *Comm. of the ACM*. 2009, Vol. 52, Issue 8, P. 36–44.
14. Andon P.I. and Balabanov O.S. (2000). Vyjavlenie znanij i izyskanija v bazah dannyh. Podhody, modeli, metody i sistemy. [Knowledge discovery and exploration in databases. Approaches, models, methods and systems]. Problems in programming. N 1–2, P. 513–526. [In Russian]
15. Balabanov O.S. (2001). Knowledge extraction from databases – advanced computer technologies for intellectual data analysis. Mathematical Machines and Systems. N 1–2. P. 40–54. [In Ukrainian]
16. Data mining: practical machine learning tools and techniques / I.H. Witten, F. Eibe, M.A. Hall. (3rd ed.). Morgan Kaufmann, San Francisco, CA. 2011. 629 p.
17. Data Mining. A Knowledge Discovery Approach. K.J. Cios, W. Pedrycz, R.W. Swiniarski and L.A. Kurgan. Springer, 2007, 606 p.
18. Azzalini A. and Scarpa B. Data analysis and Data Mining: An introduction. Oxford University Press, N.Y., 2012. 288 p.
19. Andon P.I. and Balabanov O.S. (2007). Structured statistical models: a tool for cognition and modelling. System Research and Information Technologies. N 1. P. 79–98. [In Russian]
20. Balabanov O.S. (1997). Computer's intelligence: fantastic perspectives and regular progression. Revised 2007. [In Ukrainian] [Electronic resource:] Access: https://www.researchgate.net/publication/332269445_KOMP'UTERNIJ_INTELEKT_FANTASTICNI_PERSPEKTIVI_I_SODENNIJ_POSTUP
21. Hey T, Tansley S. and Tolle K. The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research, Redmont, WA. October 2009. 252 p.
22. Siebes A. Data science as a language: challenges for computer science — a position paper. *Intern. J. of Data Science and Analytics*. 2018. Vol. 6. P. 177–187.
23. Fan J., Han F. and Liu H. Challenges of Big Data analysis. *Nat. Scient. Rev*. 2014. Vol. 1, N. 2. P. 293–314.

24. Statistical inference, learning and models in Big Data / B. Franke, J.-F. Plante, R. Roscher, E.A. Lee, C. Smyth, A. Hatefi, F. Chen, E. Gil, A.G. Schwing, A. Selvitella, M.M. Hoffman, R. Grosse, D. Hendricks and N. Reid. *Intern. Statistical Review*. 2016. Vol. 84, N 3. P. 371–389.
25. Swanson N.R. and Xiong W. Big Data analytics in economics: What have we learned so far, and where should we go from here? *Canadian Journal of Economics*. 2018. Vol. 51, Issue 3. P. 695–746.
26. The anatomy of big data computing / R. Kune, P. K. Konugurthi, A. Agarwal, R.R. Chillarige and R. Buyya. *Software: Practice and Experience*. 2016, Vol. 46. P. 79–105.
27. Smirnova E., Ivanescu A., Bai J., Crainiceanu C.M. A practical guide to big data. *Statistics and Probability Letters*. 2018. Vol. 136. P. 25–29.
28. Shi J.Q. How do statisticians analyse big data — our story. *Statistics and Probability Letters*. 2018. Vol. 136. P. 130–133.
29. Jiang H., Chen Y., Qiao Z., Weng T. H. and Li K.C. Scaling up MapReduce-based big data processing on multi-GPU systems. *Cluster Computing*. 2015. Vol. 18, N. 1. P. 369–383.
30. Haughton D. Software packages for data mining. *Wiley StatsRef: Statistics Reference Online*. 2016. P. 1–5.
31. James G., Witten D., Hastie T. and Tibshirani R. An introduction to statistical learning with applications in R. Springer, N.Y., 2013. 426 p.
32. Graham E. and Timmermann A. Forecasting in Economics and Finance. *Annual Review of Economics*. 2016. Vol. 8. P. 81–110.
33. Liu B. Web data mining: Exploring hyperlinks, contents, and usage data. Springer-Verlag: Berlin-Heidelberg, 2011. 622 p.
34. Zafarani R., Abbasi M.A. and Liu H. Social media mining. An introduction. Cambridge University Press. 2019. 380 p.
35. Big Data Analysis: New Algorithms for a New Society. N. Japkowicz and J. Stefanowski (eds.), Springer, Switzerland. 2016. 329 p.
36. Data mining for the Internet of things: Literature review and challenges. F. Chen, P. Deng, J. Wan, D. Zhang. *Intern. Journal of Distributed Sensor Networks*. Vol. 2015. 14 p.
37. Esling P. and Agón C. Time-series data mining. *ACM Computing Surveys*. 2012. Vol. 45, Issue 1. P. 12–34.
38. Pearl J. Causality: models, reasoning, and inference. Cambridge: Cambridge Univ. Press. 2000. 526 p.
39. Spirtes P., Glymour C. and Scheines R. Causation, prediction and search. New York: MIT Press, 2001. 543 p.
40. Balabanov O.S. (2017). Knowledge discovery in data and causal models in analytical informatics. Problems in Programming. N. 3. P. 96−112. [in Ukrainian]
41. Peters J., Janzing D. and Schölkopf B. Elements of Causal Inference. Foundations and Learning Algorithms. MIT Press, Cambridge, MA, USA, 2017. 265 p.
42. Shiffrin R.M. Drawing causal inference from Big Data. *Proc. Nat. Acad. Scien. USA*. 2016. Vol. 113, N. 27. P. 7308–7309.
43. Pearl J. and Bareinboim E. External validity: From do-calculus to transportability across populations. *Statistical Science*. 2014. Vol. 29, N 4. P. 579–595.
44. Balabanov O.S. (2011). From covariation to causation. Discovery of structures of dependency in data. System Research and Information Technologies. N. 4. P. 104–118. [In Ukrainian]
45. Balabanov O.S. (2016). Reconstruction of causal networks via analysis of Markov properties. Mathematical Machines and Systems. N. 1. P. 16–26. [In Ukrainian]
46. Giudici P. Financial data science. *Statistics and Probability Letters*. 2018. Vol. 136. P. 160–164.
47. Machine learning. Special issue on applications of machine learning and the knowledge discovery process. R. Kohavi, F. Provost. (Eds.) *Machine Learning*. 1998. Vol. 30, N.2/3. P. 127–274.
48. 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, August 13–17, 2016. San Francisco, California.
49. 24th SIGKDD Conference on Knowledge Discovery and Data Mining, August 19–23, 2018. London, UK.
50. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. Vol. 521. P. 436–444.
51. Donoho D.L. 50 Years of Data Science. *Journal of Computational and Graphical Statistics*. 2017. Vol. 26, Issue 4. P. 745–766.
52. Bühlmann P. and van de Geer S. Statistics for high-dimensional data: Methods, theory and applications. Springer, 2011. 556 p.
53. Bühlmann P. and van de Geer S. Statistics for big data: A perspective. *Statistics and Probability Letters*. 2018. Vol. 136. P. 37–41.

54. Secchi P. On the role of statistics in the era of big data: A call for a debate. Ibid. P. 10–14.
55. Quarteroni A. The role of statistics in the era of big data: A computational scientist' perspective. Ibid. P. 63–67.
56. Cox D.R., Kartsonaki C., Keogh R.H. Big data: Some statistical issues. Ibid. P. 111–115.
57. James G. M. Statistics within business in the era of big data. Ibid. P. 155–159.
58. Weihs C. and Ickstadt K. Data Science: the impact of statistics. *Intern. Journal of Data Science and Analytics*. 2018. Vol. 6. P. 189–194.
59. Efron B. and Hastie T. Computer age statistical inference. Cambridge University Press, N.Y., 2016. 475 p.
60. Carmichael I. and Marron J.S. Data science vs. statistics: two cultures? *Japanese Journal of Statistics and Data Science*. 2018. Vol. 1, Issue 1. P. 117–138.

Одержано 28.03.2019

Про автора:

Балабанов Олександр Степанович,
доктор фізико-математичних наук,
провідний науковий співробітник.
Кількість наукових публікацій в українських виданнях – 60.
Кількість наукових публікацій в зарубіжних виданнях – 12.
Індекс Хірша – 6.
http://orcid.org/0000-0001-9141-9074.

***Місце роботи автора*:**

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 5263420.
E-mail: bas@isofts.kiev.ua

УДК 004.62 https://doi.org/10.15407/pp2019.02.069

Б.О. Білецький

ГОРИЗОНТАЛЬНЕ ТА ВЕРТИКАЛЬНЕ МАСШТАБУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ

В роботі розглядаються основні етапи розв'язку задач машинного навчання (з учителем) розпізнаванню образів, а саме: управління навчальними вибірками, навчання, розпізнавання. Обговорюється вплив феномену великих даних (BigData) на кожен з етапів, а також методи ефективної організації обчислень на кожному з етапів при розв'язанні зазначених задач.

Ключові слова: методи машинного навчання, розпізнавання образів, горизонтальне масштабування, вертикальне масштабування, реляційні бази даних, ACID, NoSQL, CAP-теорема.

Вступ

Методи машинного навчання розпізнаванню образів останнім часом набули значної популярності. З одного боку цьому сприяє наявність предметних областей, що не підлягають ефективному описанню класичними алгоритмами та методами програмування. Це такі задачі як розпізнавання тону текстових повідомлень, виявлення аномалій у потоках даних, розпізнавання автомобільних номерних знаків, надання рекомендацій або відновлення складних функціональних залежностей. З іншого боку популярності методів машинного навчання сприяв феномен великих даних (Big Data), що тягнув за собою накопичення об'ємних навчаючих вибірок та використання їх у машинному навчанні. Адже складні задачі розпізнавання вимагають застосування більш складних моделей для розпізнавання (тобто таких моделей, що мають більше незалежних параметрів), а навчання таких моделей вимагає обробки більшої кількості прикладів у процесі навчання.

Відомо, що світові обсяги накопичених даних зростають експоненціально та, згідно оцінок, подвоюється кожні 20 місяців [1]. Натомість обчислювальна потужність комп'ютерів також зростає експоненціально згідно закону Мура [2], що був сформульований засновником компанії Intel у 1965 році, та зафіксував подвоєння швидкодії процесорів кожні 18 місяців. Такий баланс зростання обсягів даних та зростання обчислювальної потужності комп'ютерів дозволяв ефективно обробляти накопичені дані. Однак близько 2010 року стало зрозуміло, що закон Мура перестав виконуватися. Це сталося, зокрема, і через досягнення технологічних обмежень для поточної технології виробництва процесорів. У роботі [2] представлено дані щодо експоненціального зростання тактової частоти процесорів найбільших виробників, таких як: Intel, IBM, AMD, DEC, Sun з 1985 по 2010. З наведених даних видно, що близько 2010 року зростання досягло точки насичення та припинилося.

Для багатьох існуючих програмних систем, що були розраховані на обробку неперервно зростаючих масивів даних з новою актуальністю постало питання масштабування обчислень. До таких систем належать і системи машинного навчання розпізнаванню образів. Методи машинного навчання та розпізнавання складаються з кількох стадій, на які феномен великих даних (Big Data) впливає по різному. Серед важливих стадій машинного навчання розпізнаванню образів є: накопичення та зберігання навчаючих даних, навчання розпізнаванню та розпізнавання.

1. Масштабування

Після того, як перестав виконуватися закон Мура, постала необхідність у масштабуванні систем ефективного накопичення, зберігання та обробки даних. Масштабування надалі будемо розуміти як можливість системи виконувати більше обчислень паралельно без втрати швидкодії. На прикладі баз даних масштабування

можна проінтерпретувати як здатність обробляти кілька запитів одночасно, тоді як швидкодію системи пов'язуватимемо з середнім часом обробки запиту.

Для поточної технології виробництва процесорів масового споживання запропоновано два підходи масштабування: горизонтальне та вертикальне.

Вертикальне масштабування досягається за рахунок підвищення потужності окремого обчислювального пристрою. Після припинення дії закону Мура вертикальне масштабування зводиться до збільшення кількості обчислювальних ядер у комп'ютері. Такий підхід має свої обмеження, адже кількість процесорів не може зростати необмежено хоча б в силу обмеженості фізичних розмірів комп'ютерів. Крім того вартість багатопроцесорних комп'ютерів непропорційно зростає з кількістю обчислювальних ядер, що може підтримувати система. Крім того існують і принципові обмеження для розпаралелювання обчислень, що пов'язані з послідовною структурою традиційних методів програмування.

Горизонтальне масштабування полягає у використанні великої кількості достатньо простих комп'ютерів. До цієї моделі масштабування звернулась компанія Google, що одна з перших зіткнулася з феноменом великих даних. Компанія Google вирішила, відмовитися від вертикально масштабованих серверів та використовувати комп'ютери масового вжитку у власних дата-центрах через невисоку вартість останніх. Горизонтальний підхід до масштабування значно менш обмежений у сенсі простоти додавання нових вузлів та їхньої вартості, якщо порівнювати з вертикальним підходом.

Програми побудовані на основі класичних методів програмування не починають працювати швидше за наявністю додаткових процесорів або обчислювальних вузлів. Аби скористатися перевагами масштабування необхідно належним чином розпаралелити програми. Розпаралелювання призводить до втрати детермінованості програм, які стають значно складнішими для проектування та аналізу, оскільки традиційні методи (такі як операційна семантика) розраховані на послідовне виконання операцій. Розпаралелювання алгоритмів є непростою задачею, тому все більшою популярністю користуються різні парадигми (набори рекомендацій) організації паралельних обчислень, які дозволяють описувати паралельні обчислення та надають відповідний програмний інструментарій, реалізований на різних мовах програмування.

Наприклад, такий підхід як SOA (Service Oriented Architecture) дозволяє розпаралелювати досить широкий клас обчислень на основі сервісів без станів (детермінованих функцій). Цей підхід реалізовано чи не на всіх популярних мовах програмування. Іншим прикладом може бути бібліотеки Akka, що ґрунтується на алгебрі взаємодіючих процесів та похідних формалізмах, таких як π-числення, та дозволяє описувати значно ширший клас паралельних обчислень. Реалізація цієї моделі програмування доступна на мовах програмування Scala, C# та Java.

Обмеження масштабування. Ідеальним при масштабуванні є випадок, коли продуктивність обчислювальної системи лінійно залежить від кількості обчислювальних вузлів у системі. Така ситуація називається лінійною масштабованістю, та є практично недосяжною.

Закон Амдала описує приріст швидкодії системи залежно від кількості наявних обчислювальних ядер. Приріст швидкодії $S(n)$ обчислювальної системи за рахунок використання n обчислювальних ядер визначається відношенням

$$S(n) = \frac{T_s + T_p}{T_s + \frac{T_p}{n}},$$

де T_s – час обробки послідовної складової обчислення одним обчислювачем, T_p – час обробки паралельної складової обчислення одним обчислювачем.

2. Масштабування сховищ даних

Масштабування сховищ даних є важливою проблемою при побудові систем машинного навчання, адже ефективне зберігання навчаючих даних є ключовим чинником, що впливає на ефективність роботи таких систем. Окремої уваги вимагає необхідність постійно поповнювати навчаючі дані щоб покращувати якість моделей за рахунок навчання на більш актуальних даних.

Масштабування реляційних баз даних. Традиційно для ефективного зберігання великих обсягів навчаючих даних використовуються реляційні бази даних та системи управлінні ними (СУБД). Це системи що надають користувачеві механізм зберігання та отримання даних у вигляді реляцій або таблиць. Фізично реляційні СУБД являє собою потужний вертикально масштабований сервер, що виконує запити клієнтів, рис. 1. Запити до реляційної СУБД, як правило, надходять у вигляді виразів мовою SQL, а результати повертаються у вигляді реляцій. СУБД може обробляти запити багатьох клієнтів одночасно. Серед відомих реляційних СУБД є Oracle, Microsoft SQL Server, MySQL, PostgreSQL та інші.

Для реляційних баз даних характерне зберігання даних у нормалізованому вигляді, з метою мінімізації надлишковості даних. Існує ціла градація нормальних форм, кожна з яких визначає сукупність вимог до реляцій, які дозволяють уникнути певного виду надлишковості даних. Нормалізація дозволяє ефективно зберігати реляційні дані, та водночас ускладнює їхню обробку. Адже для агрегації даних, розбитих між багатьма таблицями, необхідно виконувати додаткові операції.

Окрім нормалізації даних для реляційних баз даних характерне гарантоване виконання так званих вимог ACID:

- Атомарність (Atomicity) означає, що транзакція (послідовність операцій у запиті користувача) ніколи не буде виконана частково. Або всі операції виконуються успішно, або жодна з них не виконається у разі помилки хоча б однієї із операцій у транзакції.
- Узгодженість (Consistency) означає, що в результаті успішного виконання транзакції узгодженість даних не порушується. Однак під час виконання транзакції узгодженість даних може тимчасово порушуватися.

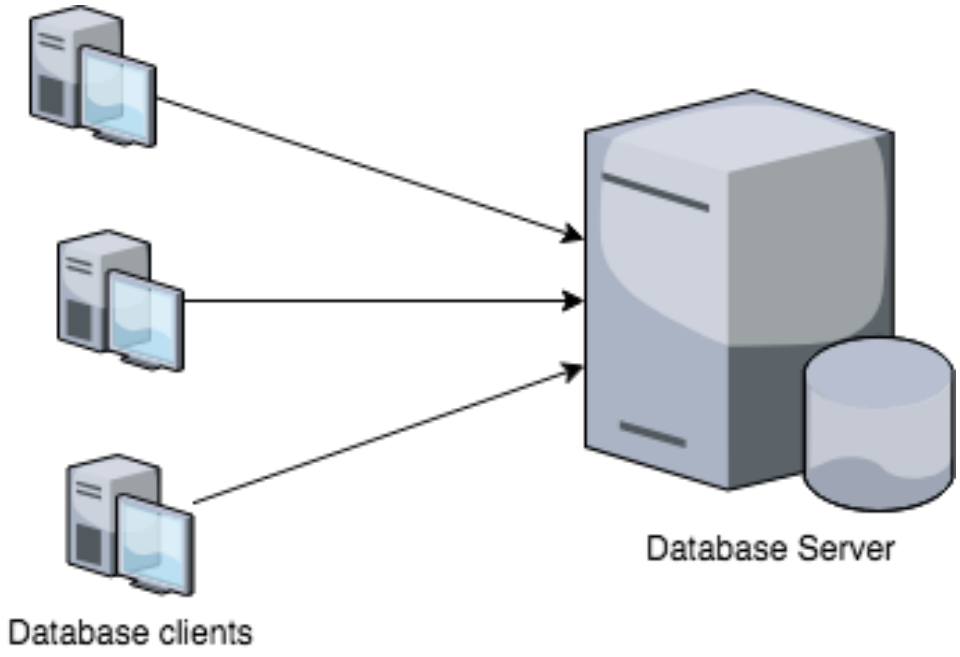


Рис. 1. Масштабування реляційних баз даних

- Ізоляція (Isolation) означає, що паралельні транзакції(обробка запитів різних користувачів) повинні бути незалежними. Ізоляція є досить складною для забезпечення вимогою, оскільки вимагає організувати ефективний доступ та модифікацію спільних ресурсів. Для забезпечення ізоляції в реляційних СУБД використовуються 2 стратегії блокування доступу до спільних ресурсів:
  - песимістичне блокування – припускає наявність колізій при паралельній роботі з ресурсом, тому доступ до нього надається тільки одній транзакції, а всі конкуруючі за доступ до ресурсу транзакції вишикуються в чергу;
  - оптимістичне блокування ґрунтується на припущенні, що колізії трапляються рідко, тому перед завершенням транзакції робиться перевірка чи не змінилася версія спільного ресурсу під час виконання транзакції. Якщо версія ресурсу не змінилася, то нова версія ресурсу зберігається в базі даних, інакше транзакція відповідальна за усунення колізій.
- Стійкість (Durability) – означає що результати успішних транзакцій будуть збережені навіть після відновлення роботи СУБД в результаті збою (такого як знеструмлення системи).

Нормалізація даних разом з вимогами ACID ускладнюють масштабування СУРБД, в результаті чого такі системи, як правило, традиційно масштабуються вертикально.

Горизонтальне масштабування нереляційних (NoSQL) баз даних. Існує ціла низка горизонтально масштабованих систем управління базами даних, що надають користувачеві механізм зберігання та отримання даних, які моделюються іншими засобами ніж табулярні реляції, які притаманні для СУБД. Такі СУБД називаються NoSQL СУБД, або NoACID СУБД, оскільки функціонують за рахунок послаблення вимог ACID. NoSQL СУБД з'явилися у 1960 роках однак набули популярності на фоні феномену великих даних. Фізично NoSQL СУБД представляють собою систему пов'язаних обчислювальних вузлів, на яких зберігаються дані, та поміж якими розподіляється обробка запитів користувачів, (рис. 2).

На сьогодні виділяють наступні типи NoSQL СУБД:

- Стовпчикові NoSQL СУБД, де дані зберігаються у вигляді стовпчиків, що характеризуються унікальним ім'ям, значенням та часом останньої модифікації). Прикладом стовпчикових NoSQL СУБД є Cassandra.
- Документо-орієнтовані NoSQL СУБД – орієнтовані на роботу з неструктурованими даними, або документами. Як документи, наприклад, можуть виступати звичайні текстові файли, або документи у

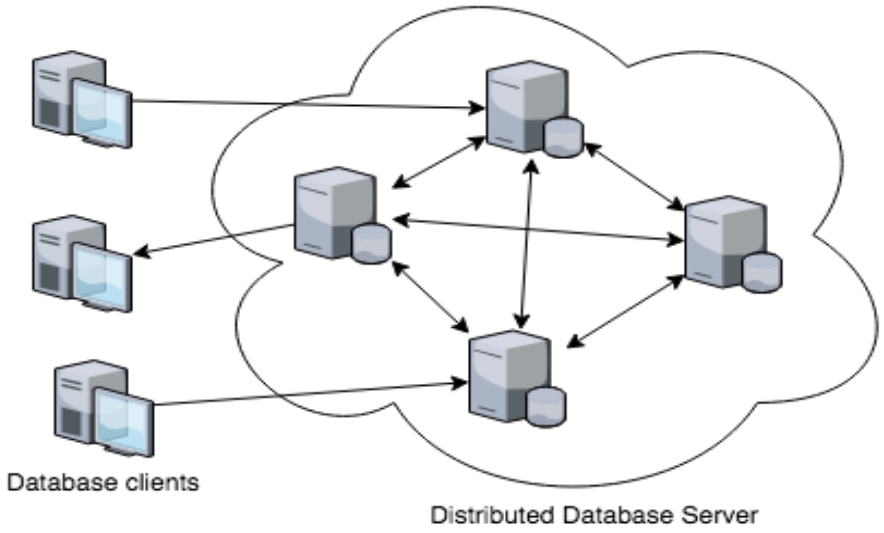


Рис. 2. Масштабування нереляційних (NoSQL) баз даних

форматі XML чи JSON. До стовпчикових відноситься така NoSQL СУБД як MongoDB.

- Ключ-значення NoSQL СУБД – дані представляються у вигляді хеш-таблиці пар ключ-значення. Ключ однозначно визначає значення, яке в свою чергу може мати внутрішню структуру та складатися з кількох елементів різних типів. До NoSQL СУБД типу ключ-значення відноситься Dynamo.

- Графові NoSQL СУБД – дані представляються у вигляді графів, тобто набору вершин та дуг, з якими пов'язані властивості різних типів. Популярною графовою NoSQL СУБД є Neo4j.

Характерними властивостями NoSQL СУБД є надлишковість при зберіганні даних, та зберігання їх у денормалізованому вигляді, а також відсутність підтримки розподілених транзакцій. Серед переваг – з'являється можливість горизонтального масштабування таких СУБД.

Обмеження горизонтального масштабування. Окрім закону Амдала існує ряд інших принципових обмежень для горизонтального масштабування.

Універсальний закон масштабування стверджує, що при збільшенні кількості обчислювальних вузлів продуктивність системи не тільки перестане зростати (згідно закону Амдала), але й взагалі знизиться. Це відбуватиметься за рахунок низки чинників, одним з яких є необхідність мережевої комунікації між вузлами в обчислювальній системі при досягнені консенсусу. Передача даних по мережі належить до операцій вводу/виводу, що є значно повільнішими за інші операції, такі як операції з оперативною пам'яттю.

CAP-теорема. Розглянемо важливі властивості розподіленої системи.

- **Узгодженість** (Consistency): всі користувачі отримують однаковий найактуальніший стан системи (або помилку), незалежно від вузла системи, що обробляє запит.

- **Доступність** (Availability): на кожен запит користувачів отримується успішна відповідь, без гарантії актуальності отриманих результатів.

- **Стійкість до втрати зв'язку** (Partition tolerance): система не втрачає працездатності та функціонує у нормальному режимі навіть за умов втрати (або затримки передачі) довільної кількості повідомлень при передачі даних між вузлами системи.

CAP-теорема стверджує, що з трьох вищенаведених властивостей в розподіленій системі одночасно досяжні не більше двох [4].

Оскільки передача даних у сучасній комп'ютерній мережі є ненадійною за своєю природою, тому у випадку розподілених СУБД неможливо знехтувати вимогою стійкості щодо втрати зв'язку (Partition tolerance), адже такі втрати завжди матимуть місце. При побудові розподілених СУБД вимога стійкості до втрати зв'язку реалізується за замовчуванням, а вибір робиться між системами, що є узгодженими (PC) або доступними (PA), рис. 3.

Популярні (PA) NoSQL СУБД гарантують послаблений варіант узгодженості системи, так звана узгодженість згодом (Eventual Consistency), що гарантує збіжність системи до узгодженого стану.

Досягнення узгодженого стану, або консенсусу, в розподілених системах стійких до обривів зв'язку є складною задачею, оскільки в таких системах не може бути фіксованої топології і, відповідно, центрального сервера, який виконував би функцію єдиного джерела актуальних стану системи. У системах, де постійно відбуваються обриви зв'язку, роль центрального сервера призначається динамічно, за допомогою алгоритмів досягнення консенсусу. Найпопулярнішими алгоритмами досягнення консенсусу, збіжність яких доведено, є Paxos та RAFT.

Окрім алгоритмів досягнення консенсусу значну роль при побудові (PA) NoSQL СУБД відіграють спеціальні структури даних CRDT (Conflict-free replicated data types), що дозволяються завжди розв'язувати неузгодженості при паралельній модифікації спільних ресурсів.

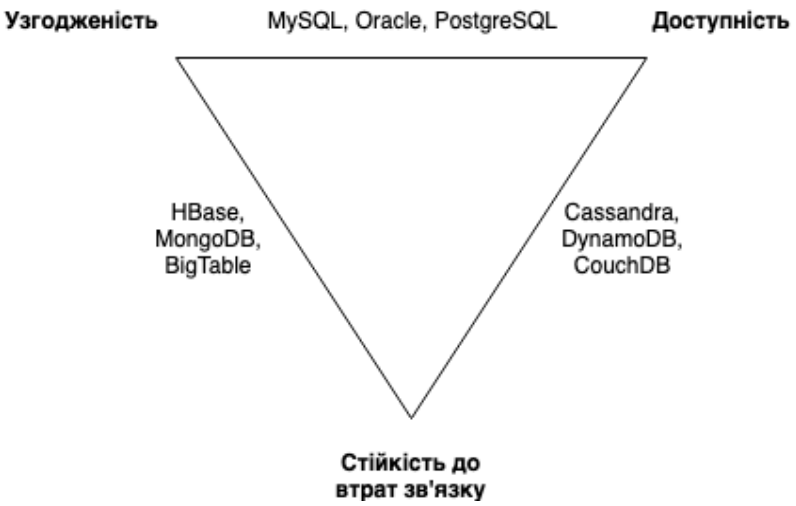


Рис. 3. CAP-теорема

3. Горизонтальне масштабування обчислень

Існує низка підходів до горизонтального масштабування обчислень. Тобто розпаралелювання обчислень за рахунок їх виконання на множині пов'язаних між собою обчислювальних пристроїв. При цьому масштабування відбувається за рахунок збільшення кількості обчислювальних вузлів системи. Горизонтальне масштабування підтримують такі методи як SOA або модель акторів, що вже згадувалися вище.

Одним з популярних підходів до масової обробки великих даних є модель програмування MapReduce що також використовує принцип горизонтального масштабування. Цей підхід надає досить простий інструментарій для описання широко класу обчислень на великих даних [5]. Простота досягається за рахунок представлення обчислень у вигляді базових операцій двох типів (Map та Reduce), що визначаються своїми алгебраїчними властивостями. Достатньо представити обчислення у вигляді композиції операцій двох зазначених типів, і реалізація моделі MapReduce бере на себе розпаралелювання відповідного обчислення, рис. 4.

При цьому дані, що обробляються, повинні зберігатися на вузлах розподіленої обчислювальної системи. При обчисленнях дані не рухаються між вузлами обчислювальної системи, а, навпаки, обчислення надсилаються до вузлів, на яких зберігаються дані для проведення обрахунків. Це

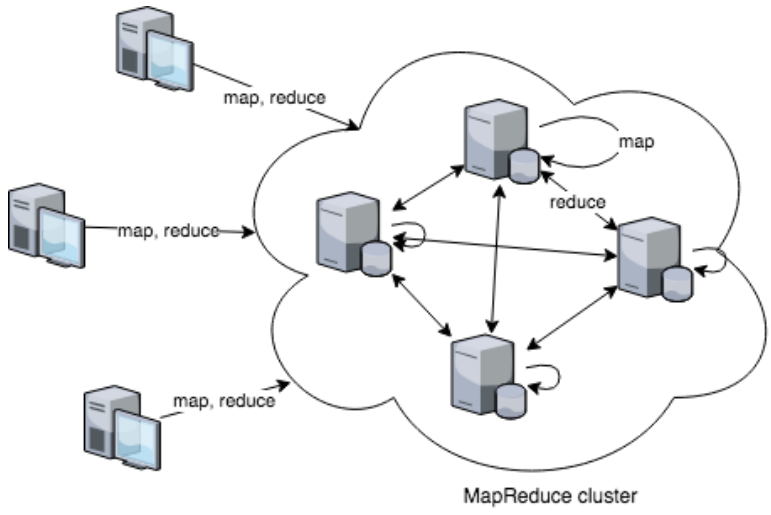


Рис. 4. MapReduce кластер з чотирьох вузлів

докорінно відрізняється від традиційного підходу Data Warehouse, за якого дані надсилаються для обробки до потужного вертикально масштабованого центрального сервера, де і проводяться обчислення.

Розглянемо основні засади, на яких ґрунтується модель програмування MapReduce. Нехай $a = (a_1, \dots, a_l)$, $a \in A^l$ – масив даних деякого типу A, розподілених між вузлами MapReduce кластера. І нехай $q: A^l \to B$ – складна процедура обробки розподіленої колекції a, що повертає результат деякого типу B. Обчислення $q(a_1, \dots, a_l)$ можна розпаралелити, якщо існує представлення

$$q(a_1, \dots, a_l) = m(a_1) \oplus \dots \oplus m(a_l),$$

де $m: A \to B$ операція обробки окремих елементів розподіленої колекції, що є детермінованою функцією, а $\oplus : B \times B \to B$ це бінарна операція, що застосовується для агрегації часткових результатів обробки масиву даних.

Операція агрегації $\oplus$ задовольняє властивостям комутативності, асоціативності та існування нейтрального елемента у множині B.

Обчислення $q(a_1, \dots, a_l)$ представлене у вигляді Map і Reduce можна зобразити у вигляді дерева, листки якого відповідають частковим результатам $m(a_1) \in B$, а вершини – операціям агрегації часткових результатів. Алгебраїчні властивості операцій Map і Reduce дозволяють перебудувати дерево обчислень зі збереженням остаточного результату обчислення таким чином, щоб мінімізувати обмін інформацією між вузлами обчислювальної системи, рис. 5.

Серед переваг підходу MapReduce є те, що у багатьох випадках він дозволяє наблизитися до лінійної масштабованості. Це досягається за рахунок мінімізації частки послідовних обчислень, що зменшує ефект закону Амдала, а також за рахунок мінімізації пересилки даних по мережі, що, в свою чергу, зменшує негативні наслідки Універсального закону масштабування.

Реалізації MapReduce. Модель програмування MapReduce є технічною специфікацією розробленою компанією Google, що обґрунтовує та надає рекомендації щодо ефективної організації масової обробки даних. Існує велика кількість реалізацій моделі MapReduce у вигляді бібліотек, або IaaS сервісів. Провайдери хмарних обчислень, такі як Google Cloud, Amazon Web Services, та Microsoft Azure дозволяють купувати кластери на основі MapReduce для масової обробки даних. Також існують провайдери спеціалізованих послуг, що фокусуються на MapReduce обчисленнях, наприклад, MapR, Databrix та Cloudera.

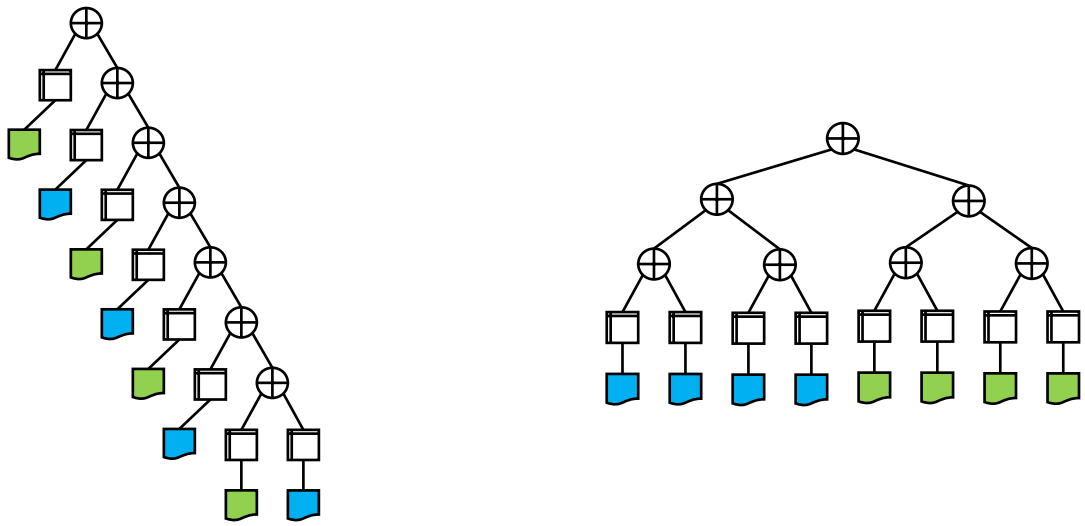


Рис. 5. Дерево обчислень MapReduce

Популярною реалізацією моделі MapReduce є бібліотека з відкритим кодом Apache Hadoop. Вона складається з двох модулів:

- з розподіленої файлової системи HDFS, що відповідає за ефективне зберігання розподілених даних;
- координатора розподілених обчислень YARN, що відповідає за координацію обчислювального навантаження між вузлами системи.

За допомогою цих двох модулів реалізується модель програмування MapReduce. Apache Hadoop є бібліотекою з відкритим кодом, тому її можна завантажити та встановити у приватній мережі комп'ютерів масового вжитку, або купити Hadoop кластер у вигляді IaaS.

Бібліотека Apache Hadoop набула значної популярності, не дивлячись на те, що базові операції Map і Reduce носять досить низькорівневий характер. Реальні обчислення складаються з композиції великої кількості таких операцій. Відповіддю на цей виклик стали більш високорівневі обчислювальні системи та бібліотеки побудовані на основі Apache Hadoop. Такі системи надають користувачеві більш високорівневі і, відповідно, більш зручні методи описання певних видів розподілених обчислень. Наприклад, існує ціла низка систем, що надають можливість виконувати SQL запити на Apache Hadoop кластерах. До таких систем належать Hive, Pig, Drill, та багато інших. Такі системи автоматично перекладають запити побудовані на мові SQL у операції Map і Reduce, та виконують їх, користуючись перевагами розподілених обчислень.

Інша відома високорівнева бібліотека, що використовує Apache Hadoop як обчислювальна модель є Apache Spark. Ця бібліотека надає зручний високорівневий інтерфейс для роботи з даними, та дозволяє зберігати проміжні дані та проводити обчислення над ними в оперативній пам'яті вузлів обчислювальної системи. Це дозволяє значно прискорити процес обчислень за рахунок уникнення чисельних звернень до накопичувачів даних, і відповідного зменшення повільних операцій вводу/виводу. Apache Spark складається з декількох модулів.

- Базового модуля Apache Spark, що реалізує базову структуру представлення даних, RDD – стійкий розподілений набір даних (Resilient Distributed Dataset), та надає високорівневі операції роботи з такими даними. Робота з розподіленими масивами даних при використанні RDD не відрізняється від роботи зі звичайними колекціями, з використанням функцій вищого порядку (тобто таких функцій, що приймають на вхід інші функції, які застосовуються до елементів колекції). Ці функції автоматично перекладаються на операції Map та Reduce та виконуються за допомогою Apache Hadoop.
- Spark Streaming – модуль, що дозволяє здійснювати розподілену обробку масивних потоків даних в реальному часі. Дані обробляються у по мірі їх надходження до обчислювальної системи. Це досягається за рахунок розбиття потоків даних у послідовність відносно невеликих RDD. Тому при роботі з потоками даних використовуються майже ті самі інтерфейси, що й при роботі з розподіленими даними.
- Spark SQL – модуль, що надає реалізацію мови запитів SQL використовуючи модель обчислень на основі RDD та Apache Hadoop. SQL – запити перетворюються на операції з RDD, які в свою чергу перетворюються на операції Map та Reduce та виконуються на Apache Hadoop.
- Spark ML – модуль, що дозволяє будувати горизонтально масштабовані розподілені методи машинного навчання, та надає цілу низку готових реалізацій різноманітних методів, серед яких: методи обробки навчаючих вибірок, методи виділення характеристик об'єктів, методи зменшення розмірності, методи навчання без учителя, методів класифікації та регресії, методи оцінки якості навчання, та багато інших.

4. Розподілені методи машинного навчання

Для з'ясування впливу феномену великих даних на методи машинного навчання розпізнаванню образів розглянемо

постановку такої задачі машинного навчання, як класифікація. Задача класифікації полягає у тому, щоб за допомогою навчаючої вибірки $\tau = (x_i, y_i)_{i=1}^{l}$ з пар $(x_i, y_i) \in X \times Y$ (характеристики об'єкта – клас), де X – множина можливих характеристик об'єкта, та Y– скінченна множина класів. Вимагається побудувати ефективну в деякому сенсі функцію класифікації $f: X \to Y$. Процедура $q(\tau)$ побудови функції класифікації $f(\cdot)$ за навчаючою вибіркою τ називається процедурою навчання.

Сама задача класифікації окремих об'єктів за наявністю побудованої в процесі навчання функції класифікації $f(\cdot)$, є, як правило, простою відносно нескладно в сенсі обчислень. Задача масової класифікації об'єктів досить просто розпаралелюється, оскільки розпізнавання окремих об'єктів відбувається незалежно, а сама функція розпізнавання є детермінованою. Горизонтальне масштабування таких обчислень є простою задачею. Обчислення такого виду можна розпаралелити за допомогою SOA, або за допомогою лише функції Map моделі програмування MapReduce.

Значно складнішою є процедура побудови функції класифікації $f(\cdot) = q(\tau)$ в процесі навчання. Навчання вимагає обробки масивних об'ємів навчаючих даних τ. Для ефективного виконання обробки навчаючих даних можна застосовувати модель програмування MapReduce. Для цього необхідно представити процедуру навчання $q(\tau)$ у вигляді операцій Map і Reduce, або описати її за допомогою більш високорівневих бібліотек. Наприклад бібліотека Apache Spark надає розподілені реалізації цілої низки методів навчання класифікації, таких як Байєсівські методи, випадковий ліс, дерево рішень, метод опорних векторів та інші.

Розподілена Байєсівська процедура розпізнавання. Розглянемо стадії навчання та класифікації на прикладі байєсівської класифікації. Байєсівський класифікатор будується на основі відомої формули Байєса у вигляді функції $f: X \to Y$

$$f_{\pi,\theta}(x) = argmax_{y \in Y} lg(\pi_x) lg(\theta_{x,y}),$$

що параметризована вектором оцінок апріорних ймовірностей класів π, а також матрицею умовних ймовірностей θ. Етап навчання $q(\tau)$ полягає у побудові вектора π та матриці θ за допомогою навчаючих даних τ. Процедуру навчання $q(\tau)$ можна розпаралелити, якщо представити її у вигляді операцій Map і Reduce.

Продемонструємо на прикладі обчислення вектора апріорних ймовірностей $\pi \in R^{|Y|}$ як виконується обробка розподіленого масиву даних під час навчання. Підрахунок елементів вектору в процесі навчання відбувається у вигляді

$$\pi_i = \frac{n_i(\tau)}{l},$$

де $n_i(\tau)$ кількість навчаючих прикладів, що потрапляють до i-го класу $i \in Y$, а l – загальна кількість навчаючих прикладів, $l = \sum_{i=1}^{n} \quad n_i(\tau)$.

Нехай Map це функція $m: X \times Y \to \to \{0,1\}^{|Y|}$, що ставить у відповідність навчаючому прикладу $(x, y) \in X \times Y$ булевий вектор розмірності $|Y|$ з одиницею на тій позиції, що відповідає порядковому номеру класу $y \in Y$, до якого потрапляє навчаючий приклад (x, y), решта елементів вектора є нульовими. Тоді операція Reduce є операцією $\oplus$ поелементної суми двох векторів розмірності $|Y|$, для таким чином визначеної операції агрегації виконуються необхідні умови асоціативності, комутативності та існування нейтрального елемента (нульовий вектор), в силу виконання всіх цих умов для операції звичайної суми цілих чисел. Після агрегації всіх часткових результатів, кожна позиція отриманого в результаті вектора відповідатиме кількості навчаючих прикладів певного класу.

Аналогічним чином підраховуються і елементи матриці θ оцінок умовних ймовірностей.

Штучні нейронні мережі та глибоке навчання. Останнім часом значної популярності набули методи глибокого навчання (Deep Learning), що використовують багатошарові штучні нейронні ме-

режі. Ці методи демонструють приголомшливі результати, особливо у розпізнаванні на неструктурованих даних. До таких задач відносяться автоматичний переклад текстів, виділення об'єктів на відео, виділення тексту зі звукових повідомлень та інші. Однією з важливих переваг таких методів є можливість автоматичного виділення характеристик об'єктів, за якими відбувається розпізнавання, з неструктурованих навчаючих даних.

Функція розпізнавання $f(x)$ у випадку штучних нейронних мереж будується у вигляді

$$f(x) = argmax_{i \in Y} a^l,$$

де σ – вектор-функція активації, a^i – збудженість нейронів i-го шару, b^i – вектор порогів активації нейронів i-го шару, а w^i матриця зв'язків нейронів i-го шару з нейронами попереднього шару.

Процедура навчання $q(\tau)$ зводиться до побудови матриць w^i та векторів b^i, що мінімізують похибку розпізнавання. Популярним методом навчання нейронних мереж є метод стохастичного градієнтного спуску, що використовує метод зворотного розповсюдження для підрахунку часткових помилок розпізнавання та для зміни відповідних коефіцієнтів матриць w^i та векторів b^i. Процедура навчання $q(\tau)$ штучних нейронних мереж заснована на зазначених методах має компактне представлення у матричній формі.

Метод градієнтного спуску є послідовною ітеративною процедурою, кожний крок якої залежить від результатів попередніх ітерацій. Навчання ускладняється глобальним характером оптимізації коефіцієнтів нейронної мережі, адже відсутність єдиності розв'язку унеможливлює агрегацію часткових результатів паралельних процесів оптимізації.

Наразі значні зусилля дослідників покладаються для розв'язання проблеми горизонтального масштабування процедур навчання штучних нейронних мереж. Одним з підходів є використання генетичних алгоритмів, за якого операція схрещування використовується для поєднання результатів, отриманих паралельними процесами оптимізації на основі стохастичного градієнтного спуску.

Масштабування штучних нейронних мереж. Слабка горизонтальна масштабованість змушує шукати альтернативні методи ефективної організації процедур навчання розпізнаванню штучних нейронних мереж. Останнім часом значної популярності набуло вертикальне масштабування за допомогою спеціалізованих графічних процесорів.

Архітектура графічних процесорів (GPU) відрізняється від архітектури традиційних процесорів (CPU).

Для традиційних CPU процесорів характерні: відносно мала кількість обчислювальних ядер (сучасні персональні комп'ютери масового вжитку мають 4 ядра), обчислювальні ядра можуть незалежно виконувати широкий спектр складних логічних операцій, відносно висока тактова частота, наявність розвиненої логіки оптимізації операцій.

Для графічних GPU процесорів навпаки, характерні: значно більша кількість обчислювальних ядер (кількість яких може сягати кількох тисяч), значно менша тактова частота, здатність виконувати паралельно лише однотипні та відносно прості операції, рис. 6.

Графічні процесори стають у нагоді для розпаралелювання масових однотипних операцій над масивами однорідних даних. До таких задач належать операції над матрицями, на яких ґрунтуються методи навчання розпізнаванню нейронних мереж. Горизонтальне масштабування за допомогою графічних процесорів реалізоване в популярній бібліотеці TensorFlow, запропонованою компанією Google [6]. Модель паралельних обчислень TensorFlow представляє обчислення у вигляді дерева, листки якого відповідають тензорам, а вузли відповідають операціям над тензорами.

Бібліотека TensorFlow є низькорівневим інструментом, тому існує низка більш високорівневих бібліотек, що спрощують розпаралелювання обчислень певного виду, використовуючи TensorFlow як

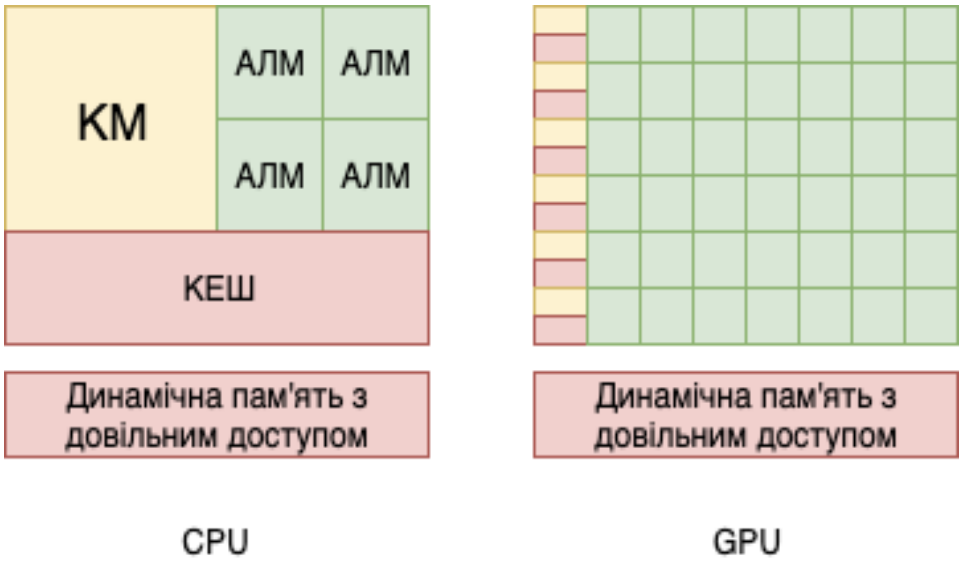


Рис. 6. Архітектура традиційного (CPU) та графічного (GPU) процесорів

обчислювальну модель. До таких бібліотек, належить Keras, що надає високорівневі методи роботи з штучними нейронними мережами.

Висновки

В роботі проаналізовано різні складові задачі машинного навчання розпізнаванню образів, а саме: 1) накопичення, збереження та обробка навчаючих даних; 2) машинне навчання; 3) розпізнавання. Було проаналізовано вплив феномену великих даних на кожну складову, а також методи масштабування та їхні обмеження. Показано, що більшість систем навчання розпізнаванню з учителем підлягають, як правило, горизонтальному масштабуванню, за виключенням методів навчання на основі нейронних мереж. Останні вимагають вертикального масштабування, оскільки використовують ітеративні алгоритми у процесі навчання.

Література

1. Hilbert M., López P. The World's Technological Capacity to Store, Communicate, and Compute Information. *Science.* 2011. Vol. 332. P. 60−65.
2. Andrew Danowitz, Kyle Kelley, James Mao, John P. Stevenson, Mark Horowitz Communications of the ACM. Vol. 55, N 4. P. 55–63.
3. Moore G. Cramming more components onto integrated circuits. *Electronics.* 1965. Vol. 38. P. 114−117
4. Seth Gilbert and Nancy Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services", ACM SIGACT News. 2002. Vol. 33, Issue 2. P. 51–59.
5. MapReduce: Simplified Data Processing on Large Clusters by Jeffrey Dean and Sanjay Ghemawat https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf
6. A computational model for TensorFlow: an introduction by Martín Abadi, Michael Isard, Derek G. Murray https://dl.acm.org/citation.cfm?doid=3088525.3088527

References

1. Hilbert M., López P. The World's Technological Capacity to Store, Communicate, and Compute Information. *Science.* 2011. Vol. 332. P. 60−65.
2. Andrew Danowitz, Kyle Kelley, James Mao, John P. Stevenson, Mark Horowitz Communications of the ACM. Vol. 55, N 4. P. 55–63.

3. Moore G. Cramming more components onto integrated circuits. *Electronics.* 1965. Vol. 38. P. 114−117
4. Seth Gilbert and Nancy Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services", ACM SIGACT News. 2002. Vol. 33, Issue 2. P. 51–59.
5. MapReduce: Simplified Data Processing on Large Clusters by Jeffrey Dean and Sanjay Ghemawat https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf
6. A computational model for TensorFlow: an introduction by Martín Abadi, Michael Isard, Derek G. Murray https://dl.acm.org/citation.cfm?doid=3088525.3088527

Одержано 18.04.2019

Про автора:

Білецький Борис Олександрович,
кандидат фізико-математичних наук,
старший науковий співробітник
Інституту кібернетики
імені В.М. Глушкова НАН України.
Кількість наукових публікацій в
українських виданнях – 13.
Кількість наукових публікацій в
зарубіжних виданнях – 2.
http://orcid.org/0000-0002-0667-8571

Місце роботи автора:

Інститут кібернетики
імені В.М. Глушкова НАН України.
03680, м. Київ,
Проспект Академіка Глушкова, 40.
Тел.: +38(044)526 2008.
E-mail: borys.biletskyy@gmail.com

УДК 004.94 https://doi.org/10.15407/pp2019.02.081

О. Захарова

ВИКОРИСТАННЯ МЕТАДАНИХ ДЛЯ ВИРІШЕННЯ ЗАДАЧ ВЕЛИКИХ ДАНИХ

Насьогодні обсяги даних, якими оперують прикладні системи, експоненціально безперервно зростають та уже давно досягли таких розмірів, що не можуть оброблятися традиційними системами. Так виник термін «Великі дані». Головні проблеми таких наборів даних пов'язані, перш за все, не лише з їх об'ємом, але й з різноманітністю, різнорідністю та складністю інформації, яку вони містять. Таким чином, разом із зростанням обсягів даних і кількості ініціатив великих даних, на перший план виходять метадані, як найважливіший пріоритет успіху проектів великих даних. Підприємства усвідомлюють, що повне використання ділового та операційного потенціалу машинного навчання, глибокого навчання та штучного інтелекту вимагає, щоб необроблені дані були доповнені метаданими. Метою даної роботи є аналіз впливу метаданих на вирішення комплексу проблем великих даних, визначення основних категорій даних, що підлягають анотуванню метаданими, та основних типів метаданих, що для цього використовуються. Насьогодні метадані є засобом класифікації, впорядкування та характеристики даних або їх вмісту. Залежно від ролі, яку вони відіграють у вирішенні задач великих даних, NISO поділяє їх на чотири типи, а саме: адміністративні, описові, структурні та мови розмітки. Різні типи метаданих можуть бути використані певним чином для ефективного вирішення задач управління, пошуку, інтеграції даних тощо. Окремим питанням є способи їх створення/автоматичної генерації, тому що ручне створення метаданих є процесом досить трудомістким, а їх обсяг часто у кілька разів перевищує обсяг самих даних.

Ключові слова: аналітика великих даних, управління великими даними, метадані, анотування, машинне навчання, Hadoop, класифікація метаданих, структурні метадані, описові метадані, адміністративні метадані, інтеграція даних, онтології, зв'язані дані, семантики даних.

Вступ

Головні проблеми при використанні великих даних пов'язані з різнорідністю, різноманітністю та складністю інформації: величезні інформаційні активи; розрізнені інформаційні сховища, які можуть розповсюджуватися на декілька бізнес-одиниць; напівструктурований та неструктурований контент даних, що досить часто розподіляється за багатьма внутрішніми операційними системами, бізнес-застосунками, мережами, серверами та інтелектуальним обладнанням.

Організація ефективного управління постійно зростаючими обсягами структурованих даних та контентом неструктурованих даних підвищує конкурентоздатність підприємства. За думкою багатьох експертів, нереалізація такого управління згодом суттєво ускладнить обслуговування вимог клієнтів, що постійно змінюються, або навіть можливо буде коштувати таким підприємствам частки ринку. Треба навчитися мати користь з великих даних та використовувати їх міць для прийняття важливих майбутніх бізнес-рішень. Здатність ефективно організовувати та класифікувати інформацію забезпечить більшу інтелектуальність у бізнесі, надаючи можливості оперативного прийняття рішень.

Таким чином, разом із зростанням обсягів даних та кількості ініціатив великих даних, на перший план виходить цінність метаданих [1]. Метадані стають найважливішим пріоритетом успіху великих даних, та їх вплив не можна недооцінювати. Важливість метаданих [2] зростає ще й в наслідок того, що підприємства усвідомлюють, що повне використання ділового та операційного потенціалу машинного навчання, глибокого навчання та штучного інтелекту вимагає, щоб необроблені дані були доповнені метаданими. Зростання обсягів фактичних даних обумовлює існування ще більшої кількості даних, або метаданих, про використання та джерела цих фактичних даних.

Роль метаданих у вирішенні проблем великих даних

При впровадженні кожного нового проекта великих даних повинна бути можливість ідентифікувати ці великі дані. Важ-

ливою можливістю для розробки та розвитку сервісів обробки великих даних є створення комплексної програми управління метаданими підприємства.

Згідно звіту, що був опублікований International Data Corporation (IDC), метадані є одним з найшвидше зростаючих підсегментів управління корпоративними даними. Однак, хоча метадані швидко зростають, вони все одно не встигають за швидким зростанням проектів великих даних. Цю проблему IDC називає як «прогалина великих даних» (“Big Data Gap”) [3]. Метадані можуть значно спростити та вдосконалити процеси збору, інтеграції та аналізу джерел великих даних. За відсутністю метаданих підприємства можуть втратити глибоке розуміння того, що саме можуть дати великі дані. Метадані можуть керувати всім життєвим циклом даних, процесами, процедурами, а також клієнтами або користувачами, які впливають на певну бізнес-інформацію. Метадані великих даних є основою для збору величезних обсягів даних з нових розрізнених джерел та інформаційних сховищ, перш ніж вони стануть некерованими.

Метадані, «маленькі» та великі дані

Метадані – це інформація, яка описує інші дані – «дані про дані» [3]. Це можуть бути описові, адміністративні та структурні дані. Це просте визначення використовувалося спеціалістами з обробки даних на протязі десятиріч. Тим не менш, метадані визначають атрибути, властивості та теги, які будуть описувати та класифікувати інформацію. Доречніше визначити метадані як «інформацію про дані». Метадані можуть бути представлені у вигляді будь-якої кількості характеристик, що пов’язані з цінними інформаційними даними, такими як тип даних, автор, дата створення, стан робочого процесу та використання в межах підприємства. Після того, як метадані визначені, вони забезпечують цінність та призначення вмісту даних та, таким чином, стають ефективним інструментом для швидкого пошуку інформації – обов’язковою умовою для аналізу великих даних та формування звітності. Але, метадані також можуть ідентифікувати «маленькі дані», які у кінцевому підсумку забезпечують структуру того, що стає великими даними. У роботі [4] (Harvard Business Review) були визначені три головні відмінності великих та невеликих даних:

- «Великі дані» спрямовані на просунення організаційних цілей, а «Маленькі дані» допомагають людям досягти особистих цілей.
- Окремі індивіди не можуть бачити великі дані.
- Великі дані контролюються організаціями, а маленькі дані контролюються приватними особами. Компанії надають фізичним особам дозвіл на доступ до великих даних, водночас як фізичні особи, навпаки, надають організаціям доступ до маленьких даних.

Щоб усвідомити істинне значення, яке метадані вносять у великі дані, треба поглянути на визначення структури, яке допомагає знаходити дані в процесі вирішення задачі виявлення даних, а також інтерпретувати та використовувати великі дані точно і правильно.

Для структурованих даних модель метаданих є «рідною» за структурою. Структуровані джерела даних можуть забезпечити логічну структуру через легко отримувані метадані. Але у великих даних немає такої доступності «власних» метаданих, тому для розкриття їх значення використовуються метадані з зовнішніх джерел даних. Великі дані потребують обробки за допомогою певної аналітики, для побудови нових визначень метаданих. Наприклад, при використанні Hadoop для збору даних не потрібно визначати метадані під час збору даних, необхідно просто визначити унікальний ключ, щоб мати можливість звертатися до даних у випадку необхідності. Однак, у кінцевому підсумку все одно треба буде визначити метадані, й Hadoop використовує для цього HCatalog. Після того, як метадані визначені, вони можуть бути співвіднесені з метаданими, що визначені в інших традиційних (структурованих) джерелах даних, забезпечуючи загальну модель метаданих для всієї організації.

Метадані можуть зв'язувати важливу інформацію організації, зв'язуючи відповідні критерії. Це дозволяє об'єднувати зв'язками аналогічні набори даних та, з іншого боку, роз'єднувати різнорідні набори даних різних джерел великих даних. Включення для великих даних атрибутів метаданих, які мають важливе значення, до напівструктурованих даних та неструктурованого контенту робить ці набори даних більш значущими, в результаті чого непотрібна інформація може бути відхилена в процесі пошуку.

Метадані забезпечують точнішу картину даних у межах всього підприємства в цілому та належний рівень погодженості даних для аналітики великих даних та бізнес-застосунків, а також керують багатьма аспектами діяльності компаній.

Так, наприклад, дослідження Стенфордського університету показали, що метадані телефонних дзвінків розкривають значні обсяги особистої інформації без доступу до реальних голосових записів. Аналіз графіків метаданих телефонних дзвінків може виявити частоту, актуальність, силу та характер взаємовідносин між людьми.

Amazon збирає метадані з продаж та в подальшому використовує їх для надання рекомендацій клієнтам та поліпшення їх взаємодії з постачальниками. Amazon також створює метадані про продукти, що доступні іншим сайтам, які надбудовують на ньому власні сервіси, збільшуючи, тим самим, обсяги продажу через Amazon.

В основі соціальних мереж також лежать метадані. Користувачі Facebook створюють метадані, коли керують списками друзів, постять статуси чи додають описи до медіа-файлів та «лайки» до статусів друзів, розділяють раніше викладений контент і надають оригінальне відео. Відстежуючи ці дії Facebook аналізує популярні теми та просуває рекламу, яка приносить користь. Згенеровані метадані використовуються для побудови пошукового індексу та рекомендацій для контенту, що є цікавим для користувачів.

Користувачі Instagram визначають підписи до зображень, які вони викладають та до яких розділяють доступ з обліковими записами інших користувачів. Instagram використовує ці дані для вдосконалення реклами. Користувачі Twitter організують людей, яким вони слідують, у списки, постять текст та відео, використовують хештеги для визначення коментарів у твіті та пов'язують їх один з одним, ретвітять чужий контент з або без коментарів, виділяють "улюблені" твіти, керують властивостями, такими як «список популярних тем». Глибина даних про суспільство, що представлена контентом у Твітері, та його метадані призвели у 2010 році до заключення угоди про те, що Бібліотека Конгресу США буде архівувати цей цінний матеріал для досліджень [5].

Щоб краще уявити, що таке є метадані та чому це важливо, розглянемо невеличкий приклад [6] метаданих (рисунок), що пов'язані з твітом лише з 140 символів. 140 символів не представляють великих обсягів даних, однак обсяги даних вибухають, якщо зв'язати твіт з усіма метаданими, що необхідні для розуміння цих 140 символів у контексті розмови.

Наведений приклад демонструє елементи метаданих.

- Ім'я та ідентифікатор користувача, що відповів автору твіта.
- Дата та час створення твіту.
- Ім'я автора.
- Ім'я користувача.
- Біографія автора.
- URL автора.
- Місцезнаходження автора.
- Надання інформації для автора.
- Дата створення облікового запису.
- Кількість обраних, що має користувач.
- Кількість користувачів, на яких підписаний даний користувач.
- Часовий пояс та зміщення часу для даного користувача.
- Мова, обрана користувачем.
- Чи є користувач захищеним.
- Кількість підписників користувача.
- Ідентифікатор місця.
- Друкована назва цього місця.
- Тип місця.
- Країна.
- Застосунок, який відіслав твіт.

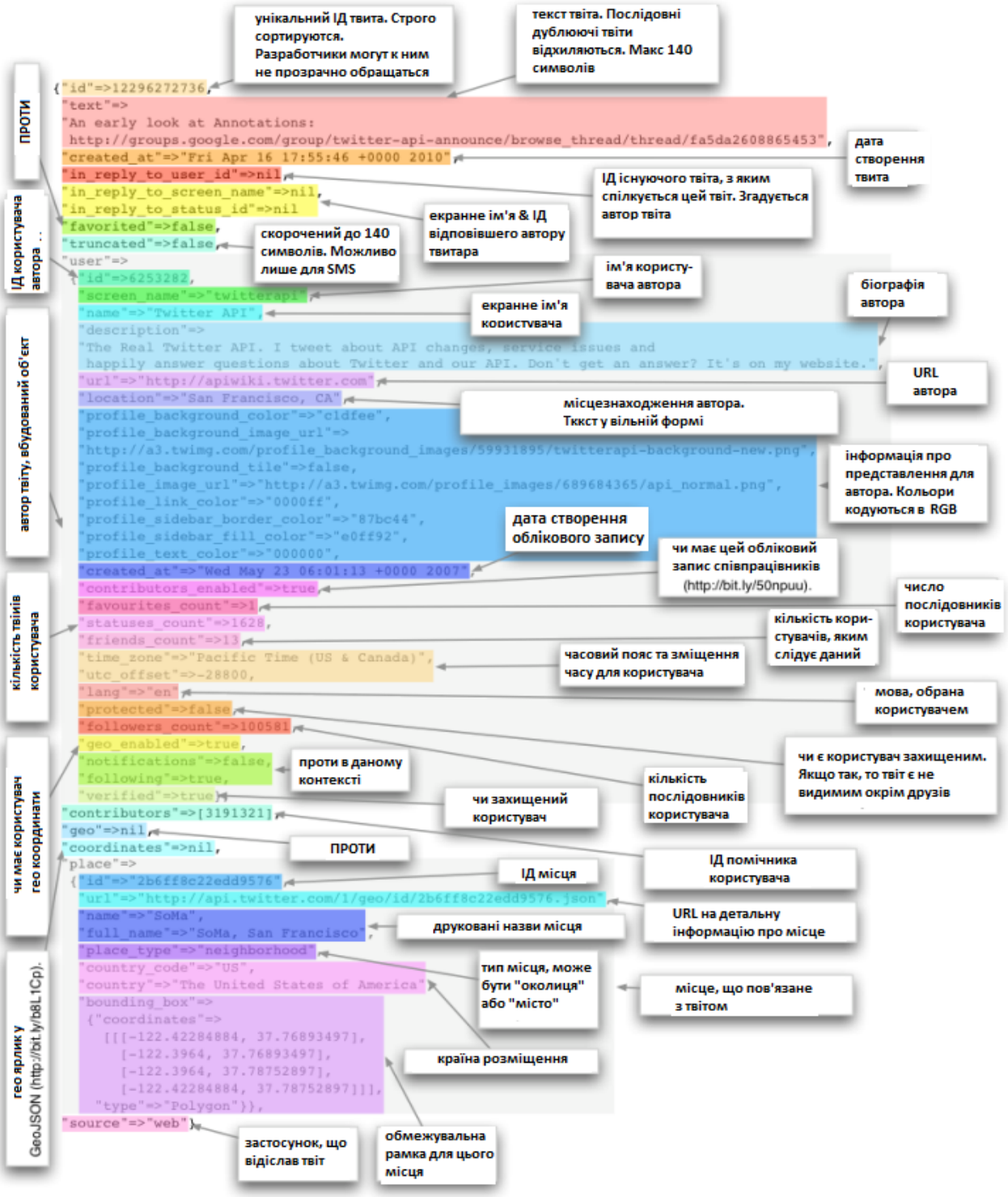


Рисунок. Метадані, пов’язані з твітом

Наведені приклади демонструють нечітку межу між метаданими та інформацією, яку вони описують. У багатьох ситуаціях ця відмінність не має значення, так як метадані часто створюються, зберігаються та обробляються так, якби це дані. Відмінністю є лише семантика.

Особливістю наведених вище прикладів є те, що всі метадані у деякій мірі структуровані. Метадані збираються таким чином, щоб вони могли виконувати корисну задачу, та сортуються за відомими категоріями. Саме це поняття структури пере-

творює необроблену інформацію у реальні метадані.

Які дані має сенс анотувати?

Очевидно, що метадані суттєво скорочують обсяги необроблених даних. Саме це відбувається, коли підприємство починає помічати (анотувати) більше своїх транзакцій та взаємодій, щоб отримати додатково уявлення про природу та контекст діалогу та взаємодії. Слід пам'ятати, що метадані при цьому стрімко збільшують обсяги даних, що зберігаються, в цілому, та не всі дані завжди є корисними для аналітики великих даних. Але деякі типи великих даних особливо підходять для аналізу, а їх анотування може суттєво вплинути на ефективне вирішення конкретних, життєво-важливих задач. Розглянемо деякі з них [6].

Відеозаписи спостережень. Загальні метадані (дата, час, місцезнаходження та ін.), як правило, автоматично прикріплюються до відеофайлу. Однак, з розповсюдженням IP-камер з'являється більше можливостей для вбудовування більшого обсягу інформації у камеру, що дозволяє аналізувати та маркувати відзнятий матеріал в режимі реального часу. Такий тип тегів може прискорити кримінальне розслідування, покращити роздрібну аналітику великих даних для моделей споживчого трафіку та покращити військову розвідку, оскільки відео з дронів у різних географічних регіонах порівнюються за кореляцією моделей (шаблонів).

Дані різного роду датчиків. У майбутньому датчики всіх типів (включаючи ті, що можуть бути імплантовані до організму людини) будуть фіксувати життєво важливі та не життєво важливі біометрії, відстежувати ефективність ліків. Це дозволить корелювати фізичну активність зі станом здоров'я, відстежувати потенційні спалахи вірусів у режимі реального часу.

Розваги та соціальні мережі. Тенденції, що засновані на великих групах людей, можуть стати відмінним джерелом великих даних, щоб допомогти вивести на ринок «наступну велику річ», допомогти обрати переможців та переможених на фондовому ринку та навіть передбачити результат виборів – все на основі інформації, яку користувачі вільно публікують через соціальні мережі.

Зображення користувачів. Люди багато розповідають про себе, коли публікують власні фото або фото сімей/друзів. Раніше картинка коштувала тисячі слів, але з появою великих даних з'явився значний множник. Ключовим моментом буде впровадження складних алгоритмів тегування, які зможуть аналізувати зображення або в режимі реального часу, коли знімки зроблені або завантажені, або в масовому порядку після їх агрегування з різних вебсайтів.

Перелічені типи даних є доповненням до звичайних ***транзакційних даних,*** що проходять через корпоративні системи в ході звичайної обробки даних.

Класифікація метаданих

Метадані є засобом класифікації, впорядкування та характеристики даних або їх вмісту. Національна організація з інформаційних стандартів (NISO – The National Information Standards Organization) [7] пропонує класифікацію [8], яка може бути застосована для всіх типів даних або репозиторіїв даних, від бібліотек до веб-сайтів, текстовых та нетекстових даних, даних у цифровій або матеріальній формі.

NISO описує наступні типи метаданих [8].

- *Описові метадані* включають таку інформацію, як, наприклад, контактні дані, заголовок або автор публікації, анотація роботи, ключові слова, географічне розміщення або навіть пояснення методології. Ці дані можуть використовуватися для виявлення, збору або групування ресурсів за загальними для них характеристиками.
- *Структурні метадані* пояснюють склад або організацію ресурсів. Наприклад, цифрову книгу можна публікувати у вигляді зображень окремих сторінок файла PDF або HTML. Ці сторінки або компоненти зазвичай групують у глави. Дані про глави, зміст або відомості про макет сторінок вважаються структурними метаданими. До структурних метаданих відносяться також такі записи, як структу-

рна карта сторінок або інших ресурсів вебсайту, подія вторгнення чи запису відомостей про голосові виклики.

– *Адміністративні метадані* використовуються для управління ресурсом. Дати створення або отримання, права доступу, права або походження, або правила утилізації, такі як зберігання чи видалення, є прикладами прав, які може застосовувати цифровий архівіст, куратор. Такі метадані є корисними для адміністратора бази даних або для адміністраторів, що відповідають за отримання даних з трафіку телекомунікаційних мереж або мереж передачі даних, або журналів систем безпеки, або даних про події.

– *Мови розмітки*. Інтегрують метадані та флаги для інших структурних чи семантичних властивостей в контенті. Змішують разом метадані та контент. Форма метаданих, що найбільш часто використовується. Вбудовані у контент флаги позначають відмічені властивості або особливості. Для текстового ресурсу це може означати маркерування структурних елементів, таких як параграфи; помітка слів семантичною інформацією – наприклад, слово є географічною назвою або певною частиною мови; також це може бути наданням інформації про форматування.

Ці різні категорії метаданих підтримують різні варіанти використання в інформаційних системах. Структурні метадані найчастіше використовуються при вирішенні задачі пошуку (виявлення) даних, так як вони дозволяють користувачам шукати або переглядати ресурси та інформацію, що їх цікавлять. Багато з властивостей метаданих доцільно відображати користувачам, щоб допомогти їм в ідентифікації або розумінні ресурсу. Вирішення задач інтероперабельності даних, ефективного обміну контентом між системами базується на метаданих, які описують цей контент. Залучені до операції системи можуть ефективно профілювати матеріал, що надходить, та співставляти його зі своїми внутрішніми структурами. Метадані підтримують управління цифровими об'єктами, надаючи інформацію, що є необхідною для належного відтворення цифрового контенту або надання відповідної версії, що задовольняє вимоги користувача.

Задача збереження великих даних може вирішуватися шляхом створення метаданих, які дозволяють перевіряти цілісність контенту після передачі даних та в інших важливих точках, а також сигналізувати у випадку виникнення необхідності вжиття заходів, що пов'язані зі збереженням даних, наприклад таких, як перенесення формату або перевірка цілісності. Нарешті, метадані підтримують навігацію частинами елементів, наприклад, від однієї сторінки або розділу до іншого, та між різними версіями об'єкта, такими як фотографічні зображення різної якості.

Таблиця далі наводить приклади елементів метаданих різних типів та варіантів їх використання в задачах великих даних.

Таблиця

Тип метаданих	**Приклади елементів метаданих**	**Варіанти використання**
Описові	Заголовок Автор Тема Жанр Дата публікації	Виявлення Візуалізація Інтероперабельність
Адміністративні	Технічні: Тип файлу Розмір файлу Дата/час створення Схема стиснення	Інтероперабельність Управління цифровими об'єктами Зберігання
	Метадані зберігання: Контрольна сума Подія збереження	Інтероперабельність Управління цифровими об'єктами Зберігання
	Правові: Статус авторського права Умови ліцензування Правовласник	Інтероперабельність Управління цифровими об'єктами
Структурні	Послідовність Місце в ієрархії	Навігація
Мови розмітки	Параграф Заголовок Список Назва Дата	Навігація Інтероперабельність

Використання метаданих для пошуку даних

Використання метаданих, що прив'язані до алгоритмів пошуку, дозволяють отримувати результати пошуку з високим ступенем достовірності. Це особливо важливо в ініціативах «Великих даних», де автономні результати, що базуються на ключових словах, можуть включати в себе скопичення менш актуальної інформації. Але використання асоціацій метаданих, створює можливість користувачам та аналітикам великих даних швидко знаходити потрібну інформацію, незважаючи на великий контент, який знаходиться у розрізнених репозиторіях.

Даний підхід може бути розповсюджений як на пошук структурованих даних, так й на пошук неструктурованого контенту репозиторіїв підприємств. Метадані можуть зв'язувати весь контент, що пов'язаний з одним або декількома атрибутами метаданих, незалежно від їх місцезнаходження або формату. Як варіант, метадані можуть надавати інформацію про елемент даних (наприклад, продукт), яка однозначно описує цей елемент. Таке поле, як ідентифікатор продукту, також є засобом для зв'язку з іншими джерелами даних з метою інтеграції даних. Окрім цього, за допомогою дескрипторів метаданих можна зв'язувати елементи даних у загальних термінах та використовувати ці метадані для інтеграції та кращого розуміння розрізнених джерел великих даних. Даний підхід надає метадані послідовно на рівні підприємства.

Дуже важливо, щоб метадані дозволяли створювати та підтримувати погодженість даних. Так, наприклад, компанії порізному визначають термін «клієнт». Та якщо компанія має розрізнені сховища даних та розкидані бізнес-одиниці, то цей термін може бути легко неправильно витлумачений по всьому підприємству і, таким чином, оцінений по-різному. Навіть, якщо кожне джерело даних визначене правильно, контекст одного й того самого елемента даних може змінюватися в різних областях застусування. Ця проблема існує в більшості організацій та, якщо її не вирішити, впливає на цілістність звітів та результати пошуку на підприємстві в цілому.

Існує два підходи до вирішення цієї проблеми:

1) перейменувати або помітити терміни застосунків, щоб вони були більш конкретними, або

2) згорнути ці імена застосунків у більш абстрактне ім'я на рівні сектора або навіть підприємства.

Саме тут буде надзвичайно корисним сховище метаданих. Адмініструючи метадані, підприємство може побудувати несуперечливе визначення або бізнес-правило для конкретного атрибуту даних та застосовувати його на рівні даних підприємства, як для структурованих, так й для неструктурованих сховищ даних.

Управління метаданими даних

Управління метаданими повинно бути частиною загальної практики управління даними підприємства. Це важливий компонент будь-якої надійної практики управління даними. Підхід, який це підтримує, полягає у встановленні управління даними для метаданих [3]. Надійні метадані забезпечують погодженість даних для підтримки підприємства та забезпечують прийняття рішень відносно аналізу великих даних. Реалізація практики управління корпоративними даними надає користувачам даних їх значення і контекст для розуміння даних та їх компонентів. Обов'язки управляючого метаданими даних включають документування контексту контенту даних (походження та спадщина даних), а також визначення даних для сутностей й атрибутів сховища даних, ідентифікацію зв'язків між даними та забезпечення перевірки своєчасності, точності та повноти даних. Підтримка належного управління метаданими сприяє успіху ініціативи великих даних та забезпечує повну реалізацію бізнес-значимості даних підприємства.

Зі зростанням обсягів використання великих даних, з'являються і будуть з'являтися надалі нові типи метаданих, які відповідають особливим вимогам різних сегментів ринку, які надають великі дані. Реалізація підходів, що засновані на метаданих, а також програми управління для

підтримки як структурованих так й великих даних, має критичне значення у встановленні загальної погодженості даних та забезпеченні кращого розуміння взаємозв'язків даних для підприємства. Реалізація відповідних корпоративних та бізнес-ініціатив дозволить досягти більшої віддачі завдяки більш швидкому доступу до конкретного контенту даних, який знаходиться у великих різноманітних сховищах великих даних, «озерах даних» [9], а також репозиторіях реляційних баз даних, що мають першорядне значення для бізнесу. Для успішного вирішення задач великих даних ключову роль відіграє вирішення проблеми їх інтеграції, в першу чергу, семантичної. Також, слід зазначити, що для управління метаданими в масштабі підприємства, необхідне створення та впровадження репозиторію метаданих.

Інтеграція даних

Розробку метаданих для різного роду прикладної інформації (біологічної, медичної та ін.) на основі стандартів семантичного веб можна розглядати як перспективний підхід для семантичної інтеграції інформації. З іншого боку, онтології, як формальні моделі для представлення інформації з чітко визначеними поняттями й взаємозв'язками між ними, можуть використовуватись для вирішення проблеми неоднорідності у джерелах даних.

Швидкий розвиток та впровадження онтологій у прикладних доменах спонукало дослідницьке товариство використовувати їх для інтеграції даних та інформації. Від моменту виникнення зв'язаних даних вони стали важливою технологією для досліджень у галузі семантики й онтологій. Багато проблем зв'язаних даних подібні проблемам великих даних, тому їх можна розглядати як частину крупнішого ландшафту великих даних. Зв'язуючий компонент зв'язаних даних приділяє особливу увагу інтеграції та об'єднанню даних у декількох джерелах. Таким чином, можна виділити три основні парадигми семантичного веб, які відіграють важливу роль у вирішенні проблем великих даних [10]: семантики, онтології та зв'язані дані.

Використання семантик

Семантичний Веб просуває стандарт анотування та інтеграції даних. Мета полягає у тому, щоб, заохочуючи включення семантичного контенту у дані, що доступні через Інтернет, перетворити існуючу мережу, де переважають неструктуровані та напівструктуровані документи, у мережу даних. Це охоплює публікацію інформації на мовах, які спеціально розроблені для даних, таких як: Resource Description Framework (RDF), Web Ontology Language (OWL), SPARQL [11] (протокол та мова запитів для джерел даних семантичного веб), і Extensible Markup Language (XML) [11]. RDF підтримує модель представлення метаданих та визначає дані трійками суб'єкт- предикат-об'єкт, що відомі як "оператори." Таке представлення гнучко пов'язує дані частинами та за принципом «посилання-за-посиланням», формуючи спрямований розмічений граф. Завдяки цьому дані та метадані тісно пов'язані один з одним, що значно спрощує пошуковий запит. Компоненти кожного RDF оператора можуть бути ідентифіковані за допомогою URI (Uniform Resource Identifiers). Окрім цього, на них можна посилатися через RDF схеми (RDFS) [12], мову веб онтологій (OWL) або інші (що не відносяться до схеми) RDF документи. Зокрема, OWL є родиною мов представлення знань для створення онтологій або баз знань. Мови характеризуються формальними семантиками та серіалізаціями на основі RDF/XML для семантичного веб. Використання інформації RDF ресурсів може здійснюватися за допомогою SPARQL (SPARQL Protocol та RDF Query Language), які здатні виявляти та обробляти дані, які зберігаються у RDF-форматі.

Онтології

Рівень онтології також має величезне значення для підтримки інтеграції даних [13]. Онтології дозволяють відображати відношення між даними, що зберігаються у базі даних. Вони надають формальне представлення набора понять

через описи базових об'єктів, класів, атрибутів та відношень. Завдяки формі представлення у вигляді дерева, онтології дозволяють зв'язувати терміни одного домена, навіть, якщо вони належать різним джерелам, в контексті інтеграції даних та ефективно співставляти різноманітні й віддалені сутності. Це дозволяє не лише покращити інтеграцію даних, але й спростити пошук інформації, а також здійснювати пошук на різних рівнях деталізації. Так, наприклад, прямий запит до терміну «рак» [14] у біомедичному контексті поверне лише дане слово у всіх входженнях, що знайдені в ресурсі. Але використовуючи специфічну онтологію (наприклад, онтологію хвороб людини – DOID [15]), результат виконання запиту буде багатшим, та включатиме такі терміни як саркома та рак, які в іншому випадку просто не будуть знайдені. Інтеграція даних на основі онтологій включає використання онтологій для ефективного об'єднання даних або інформації з декількох різнорідних джерел. Ефективність такої інтеграції багато в чому визначається погодженістю та виразністю онтологій.

Дані, що анотовані за допомогою онтологій, можуть бути інтегровані з іншими наборами даних, забезпечуючи підтримку семантик для виконання запитів. Публікація таких наборів даних як RDF, разом з їх онтологіями, забезпечує як синтаксичну так і семантичну інтеграцію даних, що була давно обіцяна технологіями семантичного веб.

Зв'язані дані

Ще одним прогресивним підходом до роботи з великими даними є парадигма зв'язаних даних [16]. Даний підхід просуває принципи гіпертексту з мережі документів у мережу «багатих» даних. Зв'язані дані описують метод для публікації структурованих даних таким чином, щоб вони були пов'язані один з одним. Це робить їх взаємозалежності більш ясними. Ця технологія заснована на семантичних веб-технологіях (зокрема, може використовувати HTTP, RDF та URI), але розширює їх для загального користування таким чином, щоб дані могли автоматично читатися IT системами.

Ідея полягає в тому, що після розміщення даних в інтернеті та визначення їх структури у машинночитаємому вигляді, їх необхідно анотувати метаданими з відкритими стандартами від W3C (наприклад, RDF та SPARQL). Це дозволить користувачам зв'язувати дані для надання контекстної інформації. Даний підхід дозволяє створювати явні зв'язки між наборами даних, використовуючи розподілені семантики зі стандартних онтологій та словників, сприяючи збільшенню ступеня інтеграції даних.

Створення метаданих

Описові метадані, як правило, створюються людиною в ручну. Деякі є просто фіксацією властивостей (назва, автор, дата видання), а деякі, такі як, наприклад, фонова інформація про автора, історія створення та ін., вимагають додаткових досліджень і лише після цього оформлення результатів. Інтерпретуюча інформація потребує залучення експертів. Спочатку технології просто дозволяли обмінюватися цими створеними в ручну метаданими. Потім почали з'являтися спеціалізовані системи введення метаданих, а потім для прискорення процесу створення метаданих стали з'являтися такі інструменти, як електронні таблиці. Останнім часом інтерфейси створення метаданих стають більш складними та зручними для користувача. Сьогодні метадані зазвичай створюються за допомогою проміжних кроків, а не безпосередньо за допомогою XML чи RDF [8]. Єдиним виключенням з цього принципу є розмітка контенту метаданими. З'являються надійні методи автоматизованого створення метаданих. Зокрема, більшість форматів файлів включають, хоча б де-яку вбудовану технічну інформацію, яка може допомогти програмному забезпеченню в інтерпретації вмісту. Також використовується системна інформація для додавання додаткової адміністративної інформації до цифрових файлів, такої як дата створення або ідентифікатор користувача, який увійшов до системи під час створення файлу. Мережеві технології та більш широка інте-

грація програмних систем полегшують реалізацію ефективного обміну метаданими, що, у свою чергу, скорочує зусилля, що витрачаються на їх дублювання.

Останнім часом з'явилися процеси для аналізу цифрового контенту та автоматичної генерації метаданих про нього. Автоматична транскрипція мови з аудіо та відео зараз є вже відносно зрілою технологією, особливо для записів, що відзняті у контролюємому середовищі з виділеними звуковими системами. Швидко розвивається технологія розпізнавання осіб для відео та нерухомих зображень. Відносно текстових ресурсів, то схований семантичний аналіз та тематичне моделювання дозволяють напівконтрольовано генерувати теми, які відносяться до текстів, що аналізуються. У дослідницьких середовищах часто використовуються технології розпізнавання частини мови й іменованих об'єктів. Розвивається автоматичне анотування зображень, що використовує алгоритми для ідентифікації об'єктів на фотографіях. Виконуються роботи з обробки сигналів для аудіофайлів, які охоплюють також створення списків відтворення в он-лайн сервісах потокової передачі музики та автоматичну класифікацію жанрів для музичних записів. Спостерігається реальний прогрес та можливості отримання високоякісних даних за допомогою програмних засобів.

Висновки

Існуючі дослідження довели, що метадані мають вирішальне значення як для загальної успішності проекту Big Data, так й для організації архітектури даних будь-якого великого підприємства.

Але, слід зазначити, що метадані суттєво збільшують і так великий об'єм даних, тому необхідно чітко розуміти, які саме дані має сенс анотувати метаданими. Інший важливий момент - метадані мають сенс, якщо вони є зрозумілими прикладним системам та людям, які їх використовують. Тут головну роль відіграють наступні моменти:

1) розробка та використання стандартів метаданих;

2) розробка та використання для створення метаданих розвинених засобів проектування, що забезпечують можливості ефективного співставлення даних;

3) окрім цього, як було зазначено, для управління метаданими в масштабі підприємства, необхідне створення та впровадження репозиторію, або сховища метаданих [6]. Насьогодні існує три підходи до його створення. Найбільш широко використовуваним нині є центральний репозиторій метаданих. Даний підхід забезпечує керовану масштабованість для захоплення нових метаданих та доступ з високою продуктивністю. Другий підхід – розподілене сховище метаданих. Воно розвивалося роками, особливо для підприємств, які мають децентралізовані бізнес-одиниці. Це дозволяє користувачам отримувати метадані з усіх сховищ у режимі реального часу. Та, нарешті, гібридний підхід використовує характеристики двох попередніх. Він підтримує доступ у режимі реального часу з інших репозиторіїв, а також забезпечує центральне джерело для підтримки визначень метаданих всього підприємства в цілому. Однак гібридний підхід розвивається досить повільно. При реалізації будь-якого з перелічених підходів потрібно враховувати семантичну інтеграцію. Незалежно від обраного підходу необхідно зв'язати різноманітний контент великих даних з самою інформацією та точно погодити правила, для яких цей контент інтерпретується. Якщо вдасться створити сховище метаданих підприємства та довести його до певного рівня зрілості, воно зможе забезпечити конкретні переваги, а саме можливість всебічного відстеження, логічні та фізичні визначення і зв'язки, міжпідприємницькі бізнес-терміни, моделі процесів, а також елементи моделі даних. Однак для інтеграції таких конструкцій метаданих необхідні спеціальні навички, та знайти потрібних спеціалістів є проблемою, яку треба вирішувати при первинному впровадженні сховища метаданих підприємства.

Література

1. https://whatis.techtarget.com/definition/metadata

2. https://www.gartner.com/doc/3075917/reasons-big-data-needs-metadata
3. https://www.datasciencecentral.com/profiles/blogs/why-you-need-metadata-for-big-data-success
4. https://hbr.org/2013/05/little-data-makes-big-data-mor
5. hts://blogs.loc.gov/loc/2010/04/how-tweet-it-is-library-acquires-entire-twitter-archive/
6. https://www.datasciencecentral.com/profiles/blogs/importance-of-metadata-in-a-big-data-world
7. http://framework.niso.org/24.html
8. https://groups.niso.org/apps/group_public/download.php/17443/understanding-metadata
9. https://www.i-scoop.eu/big-data-action-value-context/data-lakes/
10. https://groups.niso.org/apps/group_public/download.php/17443/understanding-metadata
11. "OWL Web Ontology Language Overview," W3C Recommendation, 2004, http://www.w3.org/TR/owl-features/.
12. http://www.w3.org/TR/rdf-schema/
13. Blake J. A. and Bult C. J. "Beyond the data deluge: data integration and bio-ontologies". Journal of Biomedical Informatics. 2006. Vol. 39, N 3. P. 314–320, View at Publisher · View at Google Scholar · View at Scopus.
14. Viti F., Merelli I., Calabria A. et al., "Ontology-based resources for bioinformatics analysis," International Journal of Metadata, Semantics and Ontologies. 2011. Vol. 6, N 1. P. 35–45. View at Publisher · View at Google Scholar · View at Scopus.
15. Osborne J. D., Flatow J., Holko M. et al. "Annotating the human genome with disease ontology," BMC Genomics. 2009. Vol. 10, supplement 1, article S6. View at Publisher·View at Google Scholar·View at Scopus.
16. https://www.w3.org/DesignIssues/LinkedData.html

References

1. https://whatis.techtarget.com/definition/metadata
2. https://www.gartner.com/doc/3075917/reasons-big-data-needs-metadata
3. https://www.datasciencecentral.com/profiles/blogs/why-you-need-metadata-for-big-data-success
4. https://hbr.org/2013/05/little-data-makes-big-data-mor
5. hts://blogs.loc.gov/loc/2010/04/how-tweet-it-is-library-acquires-entire-twitter-archive/
6. https://www.datasciencecentral.com/profiles/blogs/importance-of-metadata-in-a-big-data-world
7. http://framework.niso.org/24.html
8. https://groups.niso.org/apps/group_public/download.php/17443/understanding-metadata
9. https://www.i-scoop.eu/big-data-action-value-context/data-lakes/
10. https://groups.niso.org/apps/group_public/download.php/17443/understanding-metadata
11. "OWL Web Ontology Language Overview," W3C Recommendation, 2004, http://www.w3.org/TR/owl-features/.
12. http://www.w3.org/TR/rdf-schema/
13. Blake J. A. and Bult C. J. "Beyond the data deluge: data integration and bio-ontologies," Journal of Biomedical Informatics. 2006. Vol. 39, N 3. P. 314–320, View at Publisher·View at Google Scholar·View at Scopus.
14. Viti F., Merelli I., Calabria A. et al. "Ontology-based resources for bioinformatics analysis," International Journal of Metadata, Semantics and Ontologies 2011. Vol. 6, N 1. P. 35–45. View at Publisher·View at Google Scholar·View at Scopus.
15. Osborne J. D., Flatow J., Holko M. et al. "Annotating the human genome with disease ontology," BMC Genomics. 2009. Vol. 10, supplement 1, article S6, View at Publisher·View at Google Scholar·View at Scopus.
16. https://www.w3.org/DesignIssues/LinkedData.html

Одержано 19.02.2019

Про автора:

Захарова Ольга Вікторівна,
кандидат технічних наук,
старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 28.
http://orcid.org/0000-0002-9579-2973.

Місце роботи автора:

Інститут програмних систем
НАН України,
проспект Академіка Глушкова, 40.
Тел.: 526 5139.
E-mail: ozakharova68@gmail.com.
Моб. тел.: +38(068)5947560.

УДК 004.4'24

Проектирование программы визуализации земной поверхности с использованием алгебро-алгоритмического инструментария / А.Е. Дорошенко, Р.В. Кушниренко, Е.А. Яценко. – С. 3 – 10.

Одной из важных задач в рамках метеорологического прогнозирования является компьютерная визуализация полученных результатов прогноза, в частности, трехмерное моделирование рельефа земной поверхности. Одним наиболее популярных программных средств разработки приложений в области визуализации данных является OpenGL – спецификация, определяющая независимый от языка программирования кроссплатформенный программный интерфейс для написания приложений, использующих двумерную и трехмерную графику. Визуализация данных является достаточно сложной задачей, поэтому актуальным является вопрос создания специальных средств автоматизации, которые позволяли бы генерировать программный код для таких задач. В статье выполнена настройка ранее созданного алгебро-алгоритмического инструментария на проектирование и синтез OpenGL программ. Автоматизированное конструирование программ выполняется с использованием высокоуровневого языка, основывающегося на системах алгоритмических алгебр Глушкова. Подход продемонстрирован на проектировании программы визуализации рельефа подстилающей поверхности для задачи прогнозирования погоды.

Ключевые слова: автоматизированное проектирование программ, алгебра алгоритмов, визуализация данных, метеорологическое прогнозирование, рельеф земной поверхности, синтез программ, трехмерное моделирование, OpenGL.

UDC 004.4'24

Design of a terrain surface visualization program using algebra-algorithmic tools / A.Yu. Doroshenko, R.V. Kushnirenko, O.A. Yatsenko. – P. 3 – 10.

One of the important challenges in meteorological forecasting is a computer visualization of prognosis data, in particular, three-dimensional modeling of terrain surface relief. One of the most popular software tools for developing data visualization applications is OpenGL, a cross-language, cross-platform application programming interface for rendering two-dimensional and three-dimensional vector graphics. Data visualization is a fairly complex task, therefore it is necessary to develop special tools intended for automated generation of source code for such tasks. The paper deals with the further development of previously developed algebra-algorithmic tools in the direction of automated design and synthesis of OpenGL programs. Automated construction of programs is implemented using a high-level language based on Glushkov's system of algorithmic algebra. The approach is illustrated on designing a terrain surface relief visualization program for weather forecasting task.

Key words: automated software design, algorithm algebra, data visualization, meteorological forecasting, terrain surface relief, software synthesis, three-dimensional modeling, OpenGL.

УДК 004.4'24

Метод настраивания программ на платформе .Net с помощью переписывающих правил / Т.А. Мамедов, А.Ю. Дорошенко. – С. 11 – 16.

Разработано программное средство для оптимизации вычислений, которое позволяет в полуавтоматическом режиме осуществить оптимизацию программы, увеличив её быстродействие. Для этого разработан специальный плагин на систему TermWare, в котором реализована концепция правил переписывания. Плагин осуществляет работу анализатора языка C# с помощью платформы Roslyn и генерирует термы для системы TermWare с исходного кода программы. Программное средство проиллюстрировано на известном академическом примере «Игра жизнь». Приведено сравнение результатов обработки реализации алгоритма с помощью системы TermWare и библиотеки Eazfuscator.NET. Результаты были получены в следствии экспериментов на различных размерах данных. Измерена скорость работы программ до и после модификаций, количество необходимых модификаций в исходном коде для работы с системой. Испытания разработанного метода и библиотеки Eazfuscator.NET были проведены на персональном компьютере.

Ключевые слова: клеточный автомат игра «Жизнь», автотюнинг, анализатор, измерение быстродействия.

UDC 004.4'24

A method of tuning programs on .Net platform with rewriting rules / T.A. Mamedov, A.Yu. Doroshenko. – P. 11 –16.

Software tool for computation optimization that allows to optimize semiautomatically to increase execution speed was developed. Plugin for TermWare system, which implements the concept of rewriting rules, was developed. The plugin represents an analyzer for C# using Roslyn platform and generates terms for TermWare system from source code. The work of software tool was illustrated on «Conway's Game of Life» - famous academic example. A comparison of the results of processing the implementation of the algorithm using the TermWare system and the Eazfuscator.NET library was given. The results were obtained by experiments on various data sizes. Speed of the program before and after modifications, the number of modifications required in the source code to work with the system was measured. The developed system and Eazfuscator.NET framework were tested on personal computer.

Key words: cellular automaton «Conway's Game of Life», autotuning, analyzer, speed measurement.

УДК 004.853, 004.55

Проблемы использования онтологического анализа для представления знаний в Wiki-ресурсах / Ю.В. Рогушина. – С. 17 – 37.

Проанализированы направления применение Wiki-технологий для со-

UDC 004.853, 004.55

Problems of ontological analysis use for knowledge representation in Wiki-resources / J. Rogushina. – P. 17 – 37.

The main direction of the Wiki-technologies use for development of in-

здания информационных ресурсов большого объема и сложной структуры, ориентированных на функционирование в открытой среде Web. Обоснована потребность в интеллектуализации Wiki-технологии, а именно – целесообразность использования Semantic MediaWiki, для получения необходимой выразительной мощности для представления семантики основных элементов Wiki-ресурса, их свойств и связей между ними. Рассмотрена связь между представлением знаний в Semantic MediaWiki и стандартами Semantic Web, которые используются для интеграции интеллектуальных отношений и баз знаний в Web. Продемонстрирована на примерах целесообразность применения онтологического анализа и построения онтологической модели для формального и однозначного отображения структуры и контента базы знаний онлайн-версии Большой украинской энциклопедии (е-ВУЕ). Предложены методы и средства применения этой онтологической модели для создания контента е-ВУЕ и более эффективного поиска и навигации в этом информационном ресурсе.

Ключевые слова: онтология, Semantic Web, Wiki-ресурс, Wiki-онтология.

УДК 004.82

Методи і засоби отримання інформації про персоналії з авторефератів дисертацій / К.О. Кудім, Г.Ю. Проскудіна. – С. 38 – 46.

В роботі розглянуті підходи до вирішення завдання збору та вилучення розрізнених даних про персоналії з слабоструктурованих і неструктурованих документів, представлених в загальнодоступних каталогах авторефератів дисертацій. На мові PHP із застосуванням XPath розроблена система, яка дозволяє автоматично збирати первинні документи з електронного каталогу Національної бібліотеки України ім. В. І. Вернадського, ви-

formation resources of large volume and complex structure, focused on functioning in the open Web environment is analyzed. We substantiate the demand for intellectualization of the Wiki technology and compare various approaches for its semantic expansion. Such evaluation shows the expediency of the Semantic MediaWiki use to obtain the necessary expressive power to represent the semantics of the main elements of a Wiki resource, their properties and the relations between them. The relations between the representation of knowledge in the Semantic MediaWiki and the standards of the Semantic Web, which are used for integration of the Web-oriented intelligent software and knowledge bases are considered. The examples deal with the online version of the Big Ukrainian Encyclopedia (e-VUE) demonstrate the feasibility of us of ontological analysis and development of ontological model for the formal and unambiguous representatio of the structure and content of the knowledge base of the complex Wiki-resource. Methods and means of application of this ontological model for development of e-VUE content and more efficient retrieval and navigation in this information resource are proposed.

Key words: ontology, Semantic Web, Wiki-resource, Wiki-ontology.

UDC 004.82

Methods and tools for extracting personal data from theses abstracts / K.A. Kudim, G.Yu. Proskudina. – P. 38 – 46.

The problem of extraction of data about a person from scarce data collection is studied. The data collections are public resources on the internet. When these data are collected and parsed they present additional value for users. Collecting such data is problematic because of it's weak structure restrictions. Thus the system is suggested to automate information gathering and parsing. The initial task is to process personal data from thesis documents publicly available on the internet.

тягувати з цих документів дані і зберігати їх в локальному сховищі. Для зберігання обрана модель даних RDF з урахуванням особливостей даних і можливістю подальшого представлення в семантичній мережі.

Ключові слова: вилучення даних, слабоструктуровані документи, технологія XPath, регулярні вирази, семантичний веб.

This data presents information about scientists which can't be obtained from other sources. The goal is to be able to make requests to the data having its semantics in mind and not only plain text.
The prototype system is developed with PHP and XPath able to collect raw documents from digital repository of National Library of Ukraine by V. I. Vernadskiy. The system also extracts data from the collected documents and stores them locally in RDF data model suitable for specific data and for future exposition to the Semantic Web. The collection of more than 63000 documents was processed to test the system.

Key words: data extraction, weakly structured documents, XPath technology, regular expressions, semantic web.

УДК 004.855:519.216

UDC 004.855:519.216

Аналитика больших данных: принципы, направления и задачи (обзор) / А.С. Балабанов. – С. 47 – 68.

Освещены основные направления, задачи и типы результатов анализа больших (компьютеризованных) данных. Показано практическое значение больших данных и большой аналитики как фундамента создания новых компьютерных технологий планирования и управления в бизнесе. Выделены специфичные для больших данных режимы использования данных (или роды заданий анализа): «интеллектуальный» поиск нужной информации; массированная переработка («отработка») данных; индукция модели объекта (среды); экстракция знаний из данных (открытие структур и закономерностей). Очерчено этапы и организацию цикла работ по анализу данных. К типовым классам задач большой аналитики относятся: группирование случаев (кластеризация); вывод целе-определенных моделей (классификация, регрессия, распознавание); вывод генеративных моделей; выявление структур і закономерностей. Охарактеризовано особенности «глубокого обучения» и факторы его популярности. Выделены каузальные

Big Data Analytics: principles, trends and tasks (a survey) / O.S. Balabanov. – P. 47 – 68.

We review directions (avenues) of Big Data analysis and their practical meaning as well as problems and tasks in this field. Big Data Analytics appears a dominant trend in development of modern information technologies for management and planning in business. A few examples of real applications of Big Data are briefly outlined. Analysis of Big Data is aimed to extract useful sense from raw data collection. Big Data and Big Analytics have evolved as computer society's response to the challenges raised by rapid grows in data volumes, variety, heterogeneity, velocity and veracity. Big Data Analytics may be seen as today's phase of researches and developments known under names 'Data Mining', 'Knowledge Discovery in Data', 'intelligent data analysis' etc. We suggest that there exist three modes of large-scale usage of Big Data: 1) 'intelligent information retrieval; 2) massive "intermediate" data processing (concentration, mining), which may be performed during one or two scanning; 3) model inference from data; 4) knowledge discovery in data. Stages in data analysis cycle are outlined. Be-

сети как класс моделей, которые объединяют в себе преимущества генеративных, целе-определенных и многоцелевых моделей и отличаются тем, что пригодны для прогноза эффектов управления (вмешательства). Указано шесть «опор», на которых стоит методологическое ядро большой аналитики.

Ключевые слова: большие данные, анализ данных, вывод модели, открытие знаний, статистические методы, предиктивные и генеративные модели, каузальные сети, прогноз.

cause of Big Data are raw, distributed, unstructured, heterogeneous and disaggregated (vertically splitted), this data should be prepared for deep analysis. Data preparation may comprise such jobs as data retrieval, access, filtering, cleaning, aggregation, integration, dimensionality reduction, reformatting etc. There are several classes of typical data analysis problems (tasks), including: cases grouping (clustering), predictive model inference (regression, classification, recognition etc.), generative model inference, extracting structures and regularities from data. Distinction between model inference and knowledge discovery is elucidated. We give some suggestion why 'deep learning' (one of the most attractive topic by now) is so successive and popular. One of drawbacks of traditional models is they disability to make prediction under incomplete list of predictors (when some predictors are missed) or under augmented list of predictors. One may overcome this drawback using causal model. Causal networks are illuminated in the survey as attractive in that they appear to be expressive generative models and (simultaneously) predictive models in strict sense. This means they pretend to explain how the object at hand is acting (provided they are adequate). Being adequate, causal network facilitates predicting causal effect of local intervention on the object.
Methods used in Big Data Analytics will be reviewed in the next paper.

Key words: Big Data, data analysis, model inference, knowledge discovery, statistical methods, predictive and generative models, causal networks, prediction.

УДК 004.62

UDC 004.62

Горизонтальное и вертикальное масштабирование методов машинного обучения / Б. Белецкий. – С. 69 – 80.

Horizontal and Vertical Scalability of Machine Learning Methods / B. Biletskyy. – P. 69 – 80.

В работе рассматриваются основные этапы решения задачи обучения распознаванию образов, а именно: обработка и хранение обучающих данных,

The main stages of Machine Learning Pipelines are considered in the paper, such as: train data collection and storage, training and scoring. The effect of the

обучение распознаванию и распознавание. Обговаривается влияние феномена больших данных на каждый из этих этапов. Сравниваются различные подходы к эффективной организации вычислений на различных этапах. Первый раздел статьи посвящен определению понятия масштабирования, вводятся понятия горизонтального и вертикального масштабирования, обсуждаются их преимущества и недостатки. Рассматриваются некоторые ограничения при масштабировании на примере закона Амдала. Второй раздел статьи посвящен масштабированию хранилищ обучающих данных. Обговаривается подходы к масштабированию реляционных баз данных, и ограничения связанные с гарантиями ACID, которым удовлетворяют такие базы данных. Отдельно рассматриваются горизонтально масштабируемые нереляционные т. н. NoSQL базы данных. Приводится формулировка CAP-теоремы, как одного из фундаментальных ограничений при горизонтальном масштабировании таких баз данных. Третий раздел работы посвящен горизонтальному масштабированию вычислений на основе модели программирования MapReduce. Рассматриваются различные реализации этой модели программирования, такие как Hadoop и Spark, их строение и основные принципы работы. В четвертом разделе рассматриваются различные подходы к масштабированию методов машинного обучения. Приводится общая постановка задачи машинного обучения. На примере Байесовской процедуры обучения показывается, как модель программирования MapReduce применима для горизонтального масштабирования методов машинного обучения. Далее на основе глубоких нейронных сетей обговариваются методы обучения, не подлежащие горизонтальному масштабированию. Рассматриваются подходы к масштабированию таких методов при помощи графических процессоров (GPU) и модели программирования Tensor Flow.

Ключевые слова: машинное обучение, распознавание образов, горизонтальное масштабирование, вертикальное масштабирование, реляционные базы данных, ACID, NoSQL, CAP-теорема.

Big Data phenomenon on each of the stages is discussed. Different approaches to efficient organization of computation are on each of the stage are evaluated. In the first part of the paper we introduce the notion of horizontal and vertical scalability together with corresponding cons and pros. We consider some limitations of scaling, such as Amdahl's law. In the second part of the paper we consider scalability of data storage routines. First we discuss relational databases and scalability limitations related to ACID guarantees, which such database satisfy. Then we consider horizontally scalable non-relational databases, so called NoSQL databases. We formulate CAP-theorem as a fundamental limitation of horizontally scalable databases. The third part of the paper is dedicated to scalability of computation based on the MapReduce programming model. We discuss some implementations of this programming model, such as Hadoop and Spark together with some basic principles which they are based on. In the fourth part of the article we consider various approaches towards scaling of Machine Learning methods. We give the general statement of Machine Learning problem. Then we show how MapReduce programming model can be applied for horizontal scaling of Machine Learning methods on the example of Bayessian pattern recognition procedure. On the example of Deep Neural Networks we discuss Machine Learning methods which are not horizontally scalable. Then we consider some approaches towards vertical scaling of such methods based on GPU’s and the TensorFlow programming model.

Key words: machine learning, pattern recognition, horizontal scalability, vertical scalability, relational databases, ACID, NoSQL, CAP-theorem.

УДК 004.94

Использование метаданных для решения задач больших данных /
О. Захарова. – С. 81 – 91.

На сегодняшний день объемы данных, которыми оперируют прикладные системы, экспоненциально непрерывно растут и уже давно достигли таких размеров, что не могут обрабатываться традиционными системами. Так появился термин «Большие данные». Основные проблемы таких наборов данных связаны, прежде всего, не только с их объемами, но и с разнообразностью, разнородностью и сложностью информации, которую они содержат. Таким образом, вместе с ростом объемов данных та числа инициатив больших данных, на первый план выходят метаданные, как самый главный приоритет успешности проектов больших данных. Предприятия понимают, что полное использование делового и операционного потенциала машинного обучения, глубокого обучения и искусственного интеллекта требует дополнения не обработанных данных метаданными. Поэтому, целью данной работы является анализ влияния метаданных и решение комплекса проблем больших данных, определение основных категорий данных, подлежащих аннотированию метаданными, и основных типов используемых для этого метаданных. На сегодняшний день метаданные являются средством классификации, упорядочивания и характеристики данных или их содержимого. В зависимости от роли, которую они играют в решении задач больших данных, NISO выделяет четыре основных типа метаданных: административные, описательные, структурные и языки разметки. Разные типы метаданных могут использоваться определенным образом для эффективного решения задач управления, поиска, интеграции данных и т.п. Отдельным вопросом остаются способы их создания/автоматической генерации, так как создание метаданных в ручную является трудоемким процессом, а их объем за-

UDC 004.94

Using metadata to resolve big data problems / O. Zakharova. – P. 81 – 91.

Today, the volumes of data used by application systems are growing exponentially and have reached such sizes that they cannot be processed by traditional systems. So the term "Big data" appeared. The main problems of such data sets are associated, first of all, not only with their volumes, but also with the variety and complexity of the information they contain. Thus, along with the growth of data volumes and the number of big data initiatives, the metadata become the most important priority for the success of large data projects. Enterprises understand that the full use of the operational potential of machine learning, in-depth learning and artificial intellect requires the unprocessed data was supplemented with metadata. Therefore, the purpose of this work is to analyze the effect of metadata to solving the big data problems, determine the main categories of data to be annotated by metadata, and the main types of metadata used for this. Today, metadata is a means of classifying, organizing, and characterizing data or its contents. Depending on the role they play in solving big data problems, NISO identifies four main types of metadata: administrative, descriptive, structural, and markup languages. Different types of metadata can be used in a certain way to effectively solve problems of management, search, data integration, etc. A separate issue is the way of their creation/automatic generation, since the manual creation of metadata is a laborious process, and their volume is often several times larger than the volume of the data itself.

Key words: big data analytic, big data management, metadata, annotation, machine learning, Hadoop, metadata classification, structural metadata, descriptive metadata, administrative metadata, data integration, ontology, linked data, data semantics.

частую в несколько раз превосходит объем самих данных.

Ключевые слова: аналитика больших данных, управление большими данными, метаданные, аннотирование, машинное обучение, Hadoop, классификация метаданных, структурные метаданные, описательные метаданные, административные метаданные, интеграция данных, онтологии, связанные данные, семантики данных.

ДО УВАГИ АВТОРІВ!

У журналі "Проблеми програмування" публікуються наукові матеріали, які раніше не публікувалися в інших виданнях.

Мова статті: українська, російська, англійська. Обсяг статті — від 6 до 16 сторінок формату А4.

Документ зберігається у форматі doc або docx. Ім'я подається транслітерацією, як прізвище автора (авторів), наприклад, «Petrenko.doc».

Автори можуть користуватися електронною поштою і також телефаксом для ділової переписки та передачі до редакції тексту статті та правки при коректурі. E-mail редакції: tsok@isofts.kiev.ua. FAX: +380 (44) 526 6263, Телефон: 526 5065.

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 анотації має бути оформлений у 2 колонки, ширина яких – 7,86 см, а пробіл між ними – 1,27 см.

2. Послідовність розміщення та оформлення матеріалу статті.

УДК: індекс за універсальною десятковою класифікацією.

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 **(мовою статті):** 50–100 слів, не містить абревіатур, зрозумілих із змісту статті. Шрифт 10 пт, звичайний.

Ключові слова **(мовою статті):** не більше10 слів, не містить абревіатур, зрозумілих із змісту статті, подаються в називному відмінку, розділені комами. Шрифт 10 пт, звичайний.

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, має 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 спирається автор, виділення невирішених раніше частин загальної проблеми, яким присвячується дана стаття;

формулювання цілей статті (постановка задачі);

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 – 11 пт. 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.

Література англійською мовою (References): список використовуваних джерел згідно ***Harvard Style***. Джерела з заголовками на латиниці наводяться без перекладу. Для літератури джерел на мовах, що не використовують латинський алфавіт, необхідно забезпечити переведення назв джерел і вказати після них у дужках мову оригіналу. Прізвища та ініціали авторів, слід транслітерувати за правилами як для закордонного паспорта. Приклади оформлення бібліографічних посилань згідно з вимогами ***Harvard Style*** наведені в багатьох публікаціях, наприклад, за електронною адресою http://www.staffs.ac.uk/assets/harvard_referencing_examples_tcm44-39847.pdf

Дані про авторів**:** мають починатися рядком "Про авторів:", напівжирний курсив. Далі вказуються для кожного з авторів ПІБ повністю, наукове звання, посада, адреса, кількість публікацій в українських виданнях (приблизна), кількість публікацій в зарубіжних індексованих виданнях (приблизна), індекс Хірша (за наявності), 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, факс, електронна пошта, контактний телефон.

3. Оформлення файлу з анотаціями.

Файл з анотаціями містить інформацію двома мовами (наприклад, якщо стаття написана на українській мові, то анотації та ключові слова – на російській та англійській мовах) та має бути оформлений у дві колонки: УДК (шрифт – 8 пт); назва статті (шрифт – 12 пт, напівжирний); прізвища та ініціали авторів (шрифт – 12 пт); текст анотації, ключові слова (шрифт – 10 пт).

Вимоги до анотації англійською мовою: обсяг від 100 до 250 слів, інформативність, оригінальність (не є калькою української або російськомовної анотації), змістовність (відображає основний зміст статті і результати досліджень), структурованість (дотримується логіки опису результатів у статті).

Документ зберігається у форматі doc або docx. Ім'я подається транслітерацією, як прізвище автора (авторів), наприклад, «Petrenko_Annot.doc».

Примітка: Підписний індекс журналу "Проблеми програмування" – **90853**.