АЛГОРИТМ ПОШУКУ ЗВ'ЯЗКІВ І ЗАЛЕЖНОСТЕЙ У ДАНИХ ВЕБ-СТОРІНОК

УДК 004.82 004.622 519.766

КАТЕРИНИЧ Лариса Олександрівна
кандидат фізико-математичних наук, доцент, Київський національний університет імені Тараса Шевченка
Наукові інтереси: штучний інтелект, експертні системи, веб-програмування.
e-mail: katerynych@gmail.com.

ПЕТЕЛЬКО Юрій Юрійович
аспірант, Київський національний університет імені Тараса Шевченка,
Наукові інтереси: веб-програмування.
e-mail: ypetelko@gmail.com.

На даний момент існує велика кількість пошукових систем, які здійснюють тією чи іншою мірою релевантнй пошук у колекціях документів. Під терміном «документ» мається на увазі будь-який носій текстової інформації: бази даних, веб сторінки, системи електронного документообороту, тощо. Здійснюючи запит у тій чи іншій системі, користувач зазвичай отримує відповідь у вигляді списку документів, які пошукова система вважає релеватними відповідно до отриманого запиту. Існує декілька способів оцінки релевантності документа відповідно до отриманого запиту. Оцінка релевантності напряму залежить від реалізації пошукової система, яка, ґрунтуючись на деякому власному алгоритмі, вираховує відповідність вихідного документа отриманому запиту. Так, наприклад, для систем пошуку у колекціях систем електронного документообороту (зазвичай у таких системах зберігають строгі і конкретні документи юридичного чи економічного змісту) найбільше на релевантність впливає кількість входжень слів або словосполучень із запиту, а у системах веб-пошуку (Google, Bing, Yahoo), де інформація зберігається у неструктурованому вигляді, є неформалізованою, важливим фактором при визначенні релевантності документу є кількість гіперпосилань на даний документ з інших документів, відвідуваність веб-сторінки, кількість раніше здійснених запитів (схожих з даним за тими чи іншими ознаками), які були здійснені і у яких даний документ був визначений як релевантний.

Здійснивши пошук у будь-якій з таких систем, на вихід користувач отримує список документів (посилань на веб-сторінки), які система вважає релевантними. Після цього, щоб отримати важливу для користувача інформацію, потрібно відкрити (переглянути) один чи декілька з отриманих документів і проаналізувати їх. При цьому, різна інформація про об'єкт пошуку може міститися в різних документах різної релевантності і часто для того, щоб дізнатися деякий факт про об'єкт пошуку доводиться переглянути велику кількість документів для пошуку взаємозв'язку між інформацією про об'єкт що міститься в кожному з цих файлів.

Тут постає питання пошуку таких взаємозв'язків. Як сказано вище, сучасні системи пошуку просунулись відносно далеко у питаннях визначення релевантності документів, проте такі системи не здійснюють аналізу інформації для розпізнавання перехресних зв'язків.

Метою алгоритму пошуку зв'язків і залежностей (АПЗЗ) даних веб-сторінок є саме такий аналіз. Отримавши на вхід запит про деякий об'єкт, явище чи особу, система, що реалізує АПЗЗ, на вихід надає інформацію про об'єкт, а також інформацію про зв'язки цього об'єктами з іншими.

- **Визначення понять.** Перед тим як надати формальний опис всього алгоритму опишемо визначення ключовим поняттям, які використовуються при описі.

Запит (далі позначатимемо «Зп») – набір слів (формалізованих або заданих природньою мовою), які тією чи іншою мірою описують об'єкт пошуку. Поняття запиту є інтуїтивно зрозумілим. Наприклад: припустимо, що користувачу мережі інтернет (пошукової системи) потрібно знайти інформацію про «Києво-Могилянську академію». В такому разі запит може мати наступний вигляд : «Національний університет «Києво-Могилянська академія»» або «НаУКМА» або «університет Могилянка».

Експерт (користувач) – особа, яка здійснює пошук. Особа, яка здійснює пошук, зазвичай знає деякий об'єм інформації про об'єкт пошуку або цікавиться лише деякою частиною інформації про об'єкт. Для прикладу, при пошуку інформації про президента США, Біла Клінтона, після здійснення запиту «Клінтон» до пошукової системи експерт може одразу визначити ті сторінки нерелевантними, на яких міститься інформація про Хіларі Клінтон (дружину біла Клінтона).

Неважливі слова (далі позначатимемо «Нс») – слова, словоспулення, терміни і символи, які не дають ніякою корисної інформації про об'єкт пошуку. До таких слів можна віднести усі знаки пунктуації, займенники, більшість дієслів, службові частини мови, спеціальні символи розмітки (HTML, XML, тощо).

Ключові слова (далі позначатимемо «Кс») – слова і терміни, які несуть смислове навантаження і описують об'єкт пошуку. Опис об'єкту пошуку здійснюється у запиті експертом, який здійснює пошук. Тоді множина ключових слів є найбільшою підмножиною слів з множини слів запиту, яка не містить неважливих слів. Або більш формалізовано: $Кс = Зп \setminus Нс$.

Стоп-слова (далі позначатимемо «Сс») – слова та терміни, що визначаються експертом як такі, що не повністю не відповідають його запиту і при будь-яких можливих зв'язках з запитом зменшують релевантність документу, у якому зустрічається стоп-слово, або експерту неважливі зв'язки об'єкту пошуку із словом позначеним як стоп-слово. З прикладу про Клінтона, слово «Хіларі» може бути позначене як стоп-слово, якщо експерта не цікавлять зв'язки Біла Клінтона із дружиною, або слова «Афганістан» чи «війна» можуть бути позначеними як стоп-слова, якщо експерт шукає інформацію про благодійні вечори, у яких брав участь президент США.

Ресурс – документ, що містить деяку текстову інформацію. Ресурсом будемо називати будь-який текстовий документ, веб-сторінку (чи навіть веб-портал).

Пошукова система – система, що здійснює пошук ресурсів згідно заданого запиту експерта без їхнього аналізу та пошуку зв'язків. Пошуковими системами є Google, Bing, різні вузькоспеціалізовані і корпоративні системи пошуку інформації у колекціях документів.

Веб-сторінки являють собою HTML документи. Важливо зрозуміти, як саме інформація розподілена у HTML документах. Сучасні веб-сторінки – це частинки деякого веб-порталу чи веб-застосунку, що несуть велику кількість інформації і оперують дуже об'ємними HTML документами, як динамічними так і статичним. Кожен такий документ, окрім корисної для користувача інформації містить дуже багато інформації, яка є неважливою для користувача. Крім того, оскільки HTML представляє інформацію в ієрархічному вигляді і кожен елемент структурно належить тому чи іншому тегові, можна зробити висновок, що інформація, яка має сенс, а також несе деякі знання розміщується в межах одного тегу або в деякій безперервній частині документу HTML розмітки. Тобто, важлива інформація про об'єкт пошуку міститься у безпосередній близькості від ключового слова, за яким здійснювався пошук. Таким чином, можна сказати, що для знаходження зв'язків об'єкта пошуку з іншими об'єктами чи явищами, необхідно проаналізувати об'єкти, явища і терміни, які знаходяться в деякому околі від ключових слів пошуку у документі HTML. Зазвичай такі об'єкти несуть додаткову інформацію чи приховані знання про об'єкт пошуку і можуть бути використані в уточнюючих запитах до сховища даних (у нашому випадку пошукової системи), для виявлення наступних зв'язків та прихованих знань.

РОБОТА З ПОШУКОВОЮ СИСТЕМОЮ

Пошукова система є одним з ключових елементів у роботі алгоритму пошуку зв'язків і залежностей у колекціях документів і служить сховищем даних. АПЗЗ не передбачає здійснення пошуку релевантних ресурсів. Цим займається пошукова система. Пошукова система

виконує функції очищення, інтеграції і вибору даних. Робота алгоритму починається з отримання системою, що реалізує АПЗЗ (далі «система»), запиту від експерта. Запит може бути заданий як природньою мовою, так і іншими, більш формалізованими способами. Отримавши такий запит система надсилає аналогічний (або нормалізований) запит до пошукової системи, для отримання множини ресурсів, які позначаються пошуковою системою як релевантні документу. Схема взаємодії системи та пошукової системи зображена на рисунку нижче:

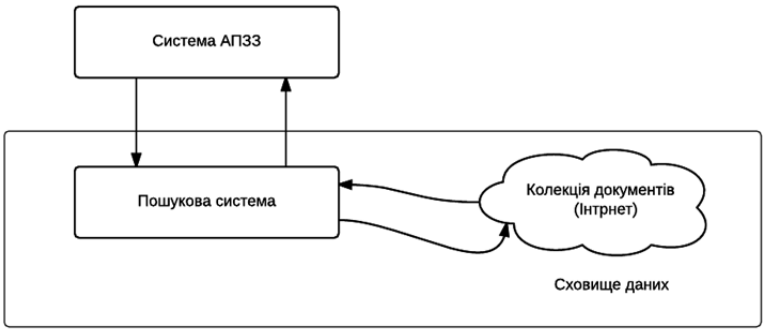

Рисунок 1.

Як видно на рисунку, робота системи (АПЗЗ) з пошуковою системою зводиться до наступного :

1. Система (АПЗЗ) отримує запит від експерта.
2. Здійснює первинну нормалізацію запиту (може не здійснювати, якщо таку нормалізацію здійснює сама пошукова система).
3. Здійснює отриманий після нормалізації (якщо нормалізація мала місце, якщо ні, то аналогічний тому, який прийшов на вхід системі) запит до пошукової системи.

У відповідь на наданий запит система отримує від пошукової системи список релевантних ресурсів (множину $D = \{ T_1, T_2, \ldots, T_n \}$), серед яких і буде здійснюватись подальший пошук зв'язків і залежностей.

Як сказано вище, якість роботи алгоритму і всієї системи прямопропорційно залежить від якості пошукової системи, яка використовується. При зменшені релевантності пошуку множини ресурсів, серед яких здійснюється пошук взаємозв'язків, якість такого пошуку буде теж зменшуватися.

НОРМАЛІЗАЦІЯ ТЕКСТОВОЇ ІНФОРМАЦІЇ

При роботі з інформацією представленою у текстовому виді важливим для будь-якого алгоритму є процес нормалізації тексту. Як описано вище, система АПЗЗ отримує запит експерта (можливі запити природньою мовою) і список релевантних ресурсів (множина D), які потрібно проаналізувати, для визначення взаємозв'язків. Оскільки запити і інформація, що зберігається на кожному ресурсі може бути представлена природньою мовою, процес нормалізації полягає у наступному:

1. Розбиття тексту на лексеми та побудова множини усіх лексем.
2. Лемматизація та стемматизація кожної лексеми.
3. Вилучення з отриманої множини усіх Нс.

Після такої нормалізації отримується множина термінів (без неважливих слів, знаків розмітки документів, знаків пунктуації), яку можна використовувати у подальшому аналізі. Важливо відзначити, що терміни у такій множині зустрічаються у такому ж порядку, як і у ресурсі, над текстом якого здійснювалася нормалізація.

Процес нормалізації є важливим, особливо, якщо пошук здійснюється мовами з розвиненою морфологією.

ПОБУДОВА МНОЖИНИ КЛЮЧОВИХ ТЕРМІНІВ

Наступним кроком після отримання списку ресурсів з релевантною інформацією, нормалізації і побудови множини термінів для кожного ресурсу є процес аналізу кожного ресурсу для пошуку взаємозв'язаних з об'єктом пошуку понять.

Перед тим, як перейти до пояснення методу відбору паттернів і зв'язаних понять, варто розібратися з тим,

як дані розміщені на веб-сторінках. Сучасні веб-сторінки мережі Інтернет містять велику кількість текстової інформації, більшість якої не несе ніякого смислового навантаження і є або рекламою, або з досить високої ймовірністю інформацією, що не має зв'язку з шуканою. Тому будувати набори частих паттернів (itemsets) для усіх термінів, що зустрічаються на ресурсі є малоефективним і часозатратним. З іншого боку, важлива смислова інформація, щодо об'єкту пошуку, зазвичай, знаходиться в деякому околі (+\- 10-20) термінів навколо ключових слів. Саме таким чином будується множина термінів, які вважаються асоційованими з термінами у запиті.

Побудова множини важливих слів здійснюється наступним чином:

1. Нехай S_{all} – множина пар <термін, кількість> усіх важливих термінів, порожня на початку аналізу.
2. Для кожного ресурсу T із множини D, будується множина термінів (позначимо її S), нормалізованих, як описано вище. Позиція кожного терміну у такій множині відома, тому до будь-якого терміну з побудованої множини можна звернутися за індексом.
3. Для кожного із термінів запиту, знаходимо його індекс у множині S .
4. Для кожного знайденого у пункту 3 індексу i, будуємо його окіл (i-n , i+n), де n – деяка наперед задана константа.
5. Для кожного індексу з околу (i-n, i+n) з множини S, додаємо термін, що знаходиться за цим індексом до множини S_{all} таким чином, що якщо термін вже існує у множині S_{all} збільшуємо значення кількості на одиницю, якщо ні – додаємо пару (термін, 1).

Після такого аналізу множина S_{all} міститиме терміни, які зустрічаються з ключовими словами найчастіше у деякому околі, що вказує на взаємозв'язок між об'єктом пошуку та знайденими термінами.

Оскільки множина S_{all} містить пари (термін, кількість) її можна відсортувати за кількістю і отримаємо терміни у порядку найчастішого вживання з ключовими словами. Чим частіше зустрічається термін у такій множині, тим сильнішим є його зв'язок з об'єктом пошуку.

Як сказано вище, пошуком релевантних даних займається сховище даних, у нашому випадку це пошукова система. Використання побудованої множини термінів S_{all} , які можуть мати зв'язок до об'єкта пошуку, для побудови уточнюючих запитів до сховища даних може мати якісний вплив на його релевантність і для виявлення нових зв'язків.

Уточнюючі запити – це повторно здійснений запит до сховища даних, у нашому випадку – до пошукової системи, у якому змінився набір ключових слів, щоб підвищити релевантність пошуку ресурсів. Як сказано вище, після першої ітерації роботи алгоритму, у розпорядженні експерта є множина термінів, пов'язаних із запитом користувача. Серед таких термів з високою ймовірністю будуть такі, що матимуть якісний вплив на результати пошуку пошуковою системою при використанні цих термінів у запиті у комбінації з попередньо вказаними ключовими словами у якості нового набору ключових слів. Після уточнюючого запиту і здійснення аналізу ресурсів таким же чином, як і на першій ітерації, множина важливих термінів поповниться новими елементами, що матимуть зв'язок з об'єктом пошуку. Ітеративний процес можна продовжувати як завгодно довго, в залежності від потреб експерта. Чим більше ітерацій буде здійснено, тим глибші зв'язки вдасться виявити на кожній наступній ітерації.

ПОВНИЙ ОПИС АЛГОРИТМУ

Розглянемо реалізація алгоритму на псевдокоді:

```
procedure FindRelationsAlgo begin define Q; # Query
define S_all; # Results set define res_list;
define res_amount = 0; define continue = TRUE; input (Q);
S_all = empty_set(); res_list = empty_list(); do
Q = normalize(Q); res_list = query_for_resources(Q ); res_amount = length(res_list);
 for resource in res_list do clean(resource); define S; define index_list; define around_set; define N;
N = 20;
S = build_term_set(resource); index_list = query_index_list(S, Q);
around_set = build_around_set(S, index_list, N);
S_all = union_of_sets(S_all, around_set); end for sort(S_all);
present_results(S_all);
```

input*(continue); if continue == TRUE define k_words;* ***input****(k_words);*

Q = ***union_of_sets****(Q, k_words); end if*

while continue == TRUE;

return S_all;

end

end procedure ***FindRelationsAlgo***

Проаналізуємо найважливіші етапи і функції алгоритму. *S_all* - множина результатів, слів та термінів, які позначаються такими, що мають зв'язок до об'єкту пошуку. Елементами множини є пари <термін, кількість>. *Q* – запит експерта (користувача), за яким здійснюється пошук.

Функція ***normalize(Q) –*** функція, у якій здійснюється нормалізація запиту користувача, виділяються ключові слова, неважливі слова прибираються, при необхідності (в залежності від пошукової системи) кожен термін запиту лемматизується і стеммінгується. На вихід функція видає список ключових слів запиту.

Функція ***query_for_resources(Q) –*** одна з важливих функцій алгоритму, що здійснює комунікацію зі сховищем даних (пошуковою системою). Отриманий на вхід список ключових слів передається пошуковій системі, після обробки запиту пошуковою системою, функція отримує відповідь пошукової систем зі списком релевантних ресурсів. Важливо зауважити, що відповідь пошукової системи може бути представлена у різних форматах (найпоширенішим є формат HTML), і дана функція повинна коректно опрацювати отриманий результат і подати на вихід лише список ресурсів.

Процедура ***clean*** *(resource)* – процедура, що здійснює початкову обробку ресурсу. Ресурси можуть бути представлені у різноманітних форматах (найпоширеніший - HTML), тому перед початком обробки і аналізу ресурсу його слід привести до правильного вигляду. У процедурі виділяється смислова (текстова) частина ресурсу, видаляються символи розмітки, знаки пунктуації, неважливі слова.

Функція ***build _term_set****(resource)* – функція, яка з отриманого на вхід тексту ресурсу подає на вихід множину (або список) термінів та слів.

Важливим є порядок термінів у множині. Слова в отриманій множині проіндексовані і розташовані в такому ж порядку, в якому вони зустрічаються в оригіналі ресурсу.

Функція ***query_index_list*** *(S, Q)* – функція, яка отримавши на вхід дві множини (два списки) термінів подає на вихід список індексів входження кожного терміну з множини *Q* у множині *S* . Тобто, в отриманому на виході списку будуть міститися індекси, за якими у множині *S* знаходяться терміни з множини *Q* .

Функція ***build_around_set*** *(S, index_list, N)* – функція, яка отримавши на вхід множину, список індексів та значення радіусу околу *N* , подає на вихід множину стермінів (слів), які знаходяться в околі *N* кожного індексу із списку *index_list.*

Реалізація функції ***build_around_set*** *(S, index_list, N)* на псевдокоді:

function ***build_around_set****(S, index_list, N)*

begin define res_list;

for index in index_list do define i = index - N;

if i < 0 i = 0;

end if

while i < index + N AND i < length(index_list) do if i != index add_to_list(res_list, S[i]);

end if

end while

end for return res_list;

end

end function build_around_set

В описані вище функції важливо звернути увагу на значення аргменту *N*. Як видно у реалізації алгоритму, аргумент ініціалізовано константою і не змінюється в процесі аналізу. Від підбору аргумента *N* залежить розмір результуючого списку. Надто мале *N* призведе до того, що кількість термінів, що потраплять у результуючу множину буде малим, і з високою ймовірністю, якість проведеного аналізу буде невисокою, оскільки велика кількість зв'язаних термінів проігнорується, не потрапивши в результуючу множину. З іншого боку, використання надто великого радіусу околу значно збільшить часові затрати роботи алгоритму, але результати суттєво не зміняться. Експериментально було визначено, що оптимальним є значення 20-35.

Функція ***union_of_sets*** *(S_all, around_set)* – функція, яка отримавши на вхід дві множини, повертає на вихід об'єднання цих множин. Варто зауважити, що

у реалізації алгоритму множина *S_all* має специфічну реалізацію. Множина *S_all* містить пари <термін, кількість>. При додаванні елемента до множини, якщо такий елемент вже існує, то у відповідній парі значення *кількість* збільшується на 1, якщо такого елемента не існує, додається нова пара <термін, 1>.

ПЕРЕВАГИ І НЕДОЛІКИ АЛГОРИТМУ, ШЛЯХИ УДОСКОНАЛЕННЯ

Описаний вище алгоритм дозволяє здійснювати ефективний пошук зв'язків між об'єктом пошуку і даними веб-сторінок. Як і кожен алгоритм у алгоритму АПЗЗ є свої переваги і недоліки.

Переваги алгоритму :

- Використання вже існуючого сховища даних. Будь-яка система, що використовує методи видобування даних і знань (Data Mining) повинна містити одним із своїх компонентів сховище даних. Сховища даних – складні інформаційні системи, розробка і підтримка яких вимагає значних часових і трудових затрат. Алгоритм АПЗЗ використовує як сховище даних будь-яку існуючу пошукову систему (або сховище даних), до яких існує доступ. Це дозволяє скоротити часові затрати і уникнути дублювання розробки уже існуючих систем.
- Мовна незалежність. Алгоритм АПЗЗ дуже мало залежить від мови, що використовується користувачем. Навіть якщо не здійснювати обробку природньої мови у процесі виконання алгоритму, результати роботи практично не змінюються. Часто обробку природніх мов здійснює сама пошукова система.
- Простота реалізації.

Недоліки алгоритму :

- Націленість на роботу з веб-сторінками. Деякою мірою алгоритм залежить від розміщення даних у ресурсі. Як сказано вище, на веб-сторінках інформація розміщується блоками і зв'язані частини інформації зазвичай знаходять недалеко одна від одної. Тому можна використовувати спосіб побудови множини важливих слів таким чином, як описано вище. Проте, інформація у інших сховищах даних може мати кардинально інше представлення, що може призвести до некоректної роботи алгоритму.
- Залежність від роботи пошукових інтернет-систем. Алгоритм залежить як від якості пошуку пошукової системи так і від швидкодії. Низька релевантність пошуку спричинить низьку якість результатів. Швидкість роботи пошукої системи і час відклику впливають на час роботи всього алгоритму. Крім того, більшість пошукових інтернет систем не дозволяють здійснювати велику кількість автоматичних запитів, що, при дуже складних аналізах може призвести до неправильного результату.
- Особливості реалізації соціальних мереж та великих порталів, на яких зберігається корисна інформація, зокрема про їхніх користувачів, часто не дозволяють здійснити аналіз описаним вище методом. Часто у відповідь на запит до таких порталів отримується закодована сторінка (наприклад JavaScript код), який не піддається аналізу. З іншого боку, кожна з великих соціальних мереж надає спеціальні інструменти для роботи з даними, що розміщені у мережі. Використання таких інструментів може якісно вплинути на роботу алгоритму, особливо при пошуку зв'язків деякої особи з іншими об'єктами, особами і даними.

Шляхи удосконалення алгоритму:

- Покращення роботи з соціальними мережами. Як сказано вище, соціальні мережі – складні веб-портали, що часто надають важливу інформацію лише з використанням певних специфічних для кожної пошукової системи інструментів. Використання таких інструментів дозволить не лише видобути множину деяких зв'язаних з об'єктом термінів, а й класифікувати її, виділити, наприклад, рік народження особи, чи її вподобання.
- Побудова дерева залежностей на кожному кроці. В описаному вище алгоритмі, кожна наступна взаємодія з пошуковою системою відкриває нові залежності. Побудова деякого дерева, на кожному рівні якого зберігатимуться нові знайдені зв'язки допоможе відповісти на питання: як саме, або через що один об'єкт має зв'язок з іншим.

ВИСНОВКИ

Алгоритм пошуку зв'язків і залежностей на вебсторінках – ефективний метод виявлення прихованих даних і зв'язків. Описаний вище алгоритм дозволяє здійснювати інтелектуальний пошук зв'язків у мережі інтернет серед даних, що знаходяться у вільному доступі.

Основними перевагами описаного алгоритму є:

- Використання вже існуючого сховища даних
- Мовна незалежність.
- Простота реалізації.

Реалізація алгоритму не вимагає трудових і часовх затрат на побудову сховища даних. Результати роботи практично не залежать від мови, якою здійснюється аналіз, користується експерт.

З іншого боку, до недоліків можна віднести:

- Алгоритм націлений на роботи з даними веб-сторінок, представлених у форматі HTML;
- Залежить від роботи пошукової системи;
- Не враховує особливостей реалізації соціальних мереж.

Алгоритм пошуку зв'язків і залежностей може мати широке практичне застосування у таких галузях:

- Пошук осіб правоохоронними органами;
- Пошук інформації про особу HR-відділами підприємств;
- Пошук зв'язків особи з іншими особами, подіями, об'єктами;
- Пошук зв'язків події, організації з іншими подіями і особами, тощо.

Використання такого алгоритму у автоматизованих системах значною мірою вплине на якість пошуку, адже на вихід отримуються не просто «сирі» веб-сторінки, а об'єкти, зв'язані з об'єктом пошуку.

ЛІТЕРАТУРА:

1. Data Mining. A Knowledge Discovery Approach / [Krzystof J. Cios, WitoldPedrych, Roman W. Swiniarski, Lukasz A. Kurgan] – San Diego, USA. : "Springer", 2007 - 606c.
2. Data Mining and Knowledge Discovery Technologies / David Taniar – New York, USA : "IGI publishing", 2007 – 369c.
3. Data Mining Patterns: New Methods and Applications / [Pascal Poncelet, Maguelonne Tesseire, Florent Masseglia] – New York, USA: "IGI publishing",
4. 2008 – 307c.
5. Data Mining with Computational Intelligence / Lipo Wang, Xiuju Fu – Berlin, Germany: "Springer", 2005 – 276c.
6. Data Mining Patterns: New Methods and Applications / [Pascal Poncelet, Maguelonne Tesseire, Florent Masseglia] – New York, USA: "IGI publishing", 2008 – 307c.
7. Data Mining. Practical Machine Learning Tools and Techniques. Second Edition / Ian H. Witten, Eibe Frank – San-Francisco, USA: "Elsevier", 2009 – 525c.
8. Learning Python, Fifth Edition / Mark Lutz – Sebastopol, USA : "O'Reilly Media Inc.", 2013 – 1540c.
9. Making Sense of Data. A Practical Guide to Exploratory Data Analysis and Data Mining / Glenn J. Myatt – Canada: "John Wiley & Sons", 2007 – 280c.
10. Pattern Recognition Algorithms for Data Mining. Scalability, Knowledge Discovery and Soft Granular Computing / Sankar K. Pal, Pabitra Mitra – Boca Raton, USA : "CRC Press", 2004 – 200c.
11. Бібліографія : [Електронний ресурс] // Вікіпедія – вільна енциклопедія – Режим доступу : http://uk.wikipedia.org/wiki/Python

Рецензент: д. ф-м.н., проф. О.І. Провотар
Київський національний університет імені Тараса Шевченка