УДК 007.3

*А.М. Глибовець*

# АЛГОРИТМ ПОШУКУ ЗВ'ЯЗКІВ І ЗАЛЕЖНОСТЕЙ МІЖ ДАНИМИ WEB-СТОРІНОК

Методи видобування й аналізу даних – відносно нова і перспективна галузь комп'ютерних наук, що знайшла своє застосування в системах інформаційному пошуку. У роботі запропоновано алгоритм пошуку зв'язків і залежностей у колекціях Web-сторінок. Алгоритм не передбачає пошуку релевантних ресурсів. Цю функцію виконує пошукова система. Вона також робить очищення, інтеграцію та вибір даних. Особливістю алгоритму є використання вже існуючого сховища даних (пошукова система або сховище даних), мовна незалежність і простота реалізації.

Ключові слова: пошукова система, PatternRecognition, алгоритм пошуку, зв'язки і залежності.

## Вступ

Задачі і методи видобування і розпізнавання Web даних (PatternRecognition) лежать на межі проблематики баз даних і штучного інтелекту. Останнім часом гостро постала потреба у добуванні лише корисної інформації і знань. З'явилася окрема галузь видобування даних Webmining, яка використовує методи DataMining для виявлення і пошуку залежностей у WWW на основі деякого «осмислення» даних [1, 2]. Традиційно виділяють чотири етапи аналізу Web даних (далі – даних): вхідний, попередньої обробки, моделювання, аналізу моделі. З'явилися такі категорії Web-Mining як аналіз використання Web ресурсів (WebUsageMining), видобування Web-структур (WebStructureMining), видобування Web-контенту (WebContent-Mining) [3].

У цій роботі, найбільшу увагу ми приділимо саме *видобуванню* Web-*контенту*. Надалі префікс Web опустимо, розуміючи його присутність із контексту видобування контенту – процес видобування знань із вмісту документів (Web-сторінок). Дані сторінок представляються у вигляді текстової, аудіо, відео інформації, або у деякому структурованому вигляді, наприклад, таблиці чи списку. Оскільки більшість такої інформації є текстовою, для її обробки варто використовувати методи *інтелектуального аналізу тексту* (IAT, TextMining або KnowledgeDiscoveryinText) [4]. Останній включає структуризацію тексту (парсинг, стеммінг), виявлення текстових паттернів, аналіз і представлення кінцевої інформації.

Ключовими задачами IAT є кластеризація текстів, обробка змін у колекціях текстів і пошук, які вирішуються на основі розпізнавання іменованих елементів (сутностей, власних назв, імен), пошуку зв'язків об'єкта, виділення термінології (знаходження ключових слів), автореферування (виділення з тексту змістовної чи оціночної інформації).

Більшість Web-документів представляють текстову інформацію у HTML форматі. Цей формат має багато спеціальних символів розмітки, за допомогою яких можна ідентифікувати корисну інформацію. Проте, навіть спеціальна розмітка Web-сторінки мало впливає на їх структурованість. Звичайний текстовий документ складається з абзаців чи параграфів, тоді як Web-сторінка складається з різних елементів розмітки таких як навігаційна панель, меню, таблиці, заголовки. Тому, стандартні методи інтелектуального аналізу тексту важко застосувати у процесі аналізу даних Web-сторінок.

Здійснюючи запит, користувач зазвичай отримує відповідь у вигляді списку документів, які пошукова система вважає релеватними відповідно до отриманого запиту. Важливим фактором при визначенні релевантності документу є кількість гіперпосилань на даний документ з інших документів, відвідуваність сторінки, кількість раніше здійснених запитів

(схожих з даними за тими чи іншими ознаками), які були здійснені і в яких даний документ був визначений як релевантний. Різна інформація про об'єкт пошуку може міститися в різних документах різної релевантності і для уточнення деякого факту доводиться переглянути велику кількість документів для пошуку взаємозв'язку між інформацією про об'єкт що міститься в кожному з цих документів. Постає задача ефективного пошуку таких взаємозв'язків. Сучасні пошукові системи суттєво просунулись у питаннях визначення релевантності документів, проте вони не мають засобів аналізу інформації для розпізнавання зазначених перехресних зв'язків.

Опишемо запропонований нами алгоритм пошуку зв'язків і залежностей (АПЗЗ) у даних Web-сторінок. Отримавши на вхід запит про деякий об'єкт, пошукова система з АПЗЗ надає на виході окрім інформації про об'єкт і інформацію про його зв'язки з іншими об'єктами.

## Основна частина

*Визначення основних понять. Запит* (далі позначатимемо «Зп») – набір слів (формалізованих або заданих природньою мовою), які описують об'єкт пошуку.

*Експерт* (користувач) – особа, яка здійснює пошук. Вона зазвичай має певні знання (уявлення) про об'єкт пошуку або його частину.

*Неважливі слова* (далі позначатимемо «Нс») – слова, словосполучення, терміни і символи, які не дають 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'єкта 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:

$$Кс = Зп \setminus Нс.$$

*Стоп-слова* (далі позначатимемо «Сс») – слова та терміни, що визначаються експертом як такі, що не повністю 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'єкта пошуку із словом позначеним як стоп-слово.

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

*Пошукова система* (ПС) – система, що здійснює пошук ресурсів згідно заданого запиту експерта без їхнього аналізу та пошуку зв'язків.

Сучасні Web-сторінки – це елементи деякого Web-порталу чи Web-застосунку, які несуть велику кількість інформації і оперують дуже об'ємними динамічними і статичними HTML документами. Оскільки HTML представляє інформацію в ієрархічному вигляді і кожен елемент структурно належить якомусь тегові, можна зробити висновок. Інформація, яка має сенс, а також несе деякі знання розміщується в межах одного тегу або в деякій неперервній частині документу HTML розмітки. Тобто, важлива інформація про об'єкт пошуку міститься у безпосередній близькості від ключового слова, за яким здійснювався пошук. Тому можна стверджувати, 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.

*Підготовча робота*

АПЗЗ не передбачає здійснення пошуку релевантних ресурсів. Цим займа-

ється ПС. Вона виконує функції очищення, інтеграції і вибору даних. Робота алгоритму починається з отримання запиту від експерта. Запит може бути заданий як природною мовою, так і іншими, більш формалізованими способами. Система реалізація алгоритму надсилає аналогічний (або нормалізований) запит до ПС, для отримання списку ресурсів, які позначаються ПС як релевантні документи. У відповідь система отримує від ПС список релевантних ресурсів ( $D=\{T_1,T_2,...,T_n\}$), серед яких і буде здійснюватись подальший пошук зв'язків і залежностей.

Зрозуміло, що якість роботи алгоритму прямо пропорційно залежить від якості видачі ПС.

Для будь-якого алгоритму обробки текстової інформації важливим є процес нормалізації тексту. Він складається із трьох кроків: розбиття тексту на лексеми та побудова множини усіх лексем; стематизації кожної лексеми; вилучення з отриманої множини усіх неважливих слів (Нс). Після нормалізації отримується множина термінів для подальшого аналізу.

Наступним кроком є аналіз кожного ресурсу для пошуку взаємопов'язаних з об'єктом пошуку понять у визначеному околі.

Побудова списку важливих слів здійснюється наступним чином:

– нехай $S_{all}$ – список пар <термін, кількість> усіх важливих термінів, порожній на початку аналізу;

– для кожного ресурсу $T$ із $D$, будується нормалізований список термінів $S$;

– для кожного із термінів запиту, знаходимо його індекс і у списку $S$;

– для кожного індексу $i$, будуємо його окіл ($i-n,i+n$), де $n$ – деяка наперед задана константа;

– 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).

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

Оскільки список $S_{all}$ містить пари (термін, кількість) його можна впорядкувати за кількістю. Чим частіше зустрічається термін у такій множині, тим сильнішим є його зв'язок з об'єктом пошуку.

Як ми зазначали раніше, пошуком релевантних даних займається ПС. Використання списку термінів $S_{all}$, для побудови уточнюючих запитів , які можуть мати зв'язок з об'єктом пошуку, може мати якісний вплив на покращення релевантності і виявлення нових зв'язків.

Уточнюючі запити – це повторно здійснений запит до ПС, у якому змінився набір ключових слів. Після першої ітерації роботи алгоритму, у розпорядженні експерта постане список термінів, пов'язаних із запитом користувача. Серед таких термів з високою ймовірністю будуть такі, що матимуть якісний вплив на результати пошуку за використанням цих термінів у новому запиті у комбінації з попередньо вказаними ключовими словами. Після уточнюючого запиту і здійснення аналізу ресурсів за тією ж процедурою, що і на першій ітерації, список важливих термінів поповниться новими елементами, які мали 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.

## Опис алгоритму

Формалізований опис алгоритму відображає наступна процедура:

```
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
```

Опишемо основні складові алгоритму.

Зрозуміло, що $Q$ – запит експерта (користувача), за яким здійснюється пошук.

Функція ***normalize*** ($Q$), здійснює нормалізацію запиту користувача, виділення ключових слів, усунення неважливих слів і за необхідності (в залежності від пошукової системи) кожен термін запиту лематизується і стемінгується. На вихід функція видає список ключових слів запиту.

*Функція* ***query_for_resources*** ($Q$) здійснює комунікацію зі сховищем даних (пошуковою системою). Отриманий на вхід список ключових слів передається ПС. Після обробки запиту пошуковою системою, функція отримує у список релевантних ресурсів. Зауважимо, що цей список може представлятися у різних форматах (найпоширенішим є формат HTML). Тому наша функція повинна коректно опрацювати отриманий результат і подати на вихід лише змістовний список ресурсів.

Процедура ***clean*** (*resource*) здійснює початкову обробку ресурсу, представленому у деякому форматі (найпоширеніший – HTML). Тому попередньо ресурс слід привести до змістовного вигляду. У процедурі виділяється смислова (текстова) частина ресурсу, видаляються символи розмітки, знаки пунктуації, неважливі слова.

Функція ***build_term_set*** (*resource*) із отриманого на вхід тексту ресурсу подає на вихід список термінів та слів. Важливим є порядок термінів у списку. Слова в отриманому списку проіндексовані і розташовані в такому ж порядку, в якому вони зустрічаються в оригіналі ресурсу.

Функція ***query_index_list*** ($S$, $Q$) отримавши на вхід два списки термінів подає на вихід список індексів входження кожного терміну з $Q$ у $S$.

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

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

```
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
```

Звернемо увагу на значення аргументу *N* . Цей параметр ініціалізовано константою і не змінюється в процесі аналізу. Від підбору аргументу *N* залежить розмір результативного списку. Надто мале *N* призведе до того, що кількість термінів, які потраплять у результативну множину буде малою, і з високою ймовірністю, якість проведеного аналізу буде невисокою, оскільки велика кількість зв'язаних термінів проігнорується, не потрапивши в результативний список. З іншого боку, використання надто великого радіусу околу значно збільшить часові затрати роботи алгоритму, але результати суттєво не зміняться. Результати тестування показали, що оптимальними є значення з проміжку 20–35.

Функція ***union_of_sets*** *(S _all, around_set)* виконує роль об'єднання результату.

Для тестування алгоритму побудовано програмну систему у вигляді клієнт-серверного Web застосунку (Web-сайту), серверна частина якого відповідала за аналіз і роботу самого алгоритму, а користувачу надавалася Web-сторінка з отриманими даними. Інтерфейсом доступу та пошуку релевантних даних служила пошукова система Google. Мовою програмування виступав *Python*.

Існує багато OpenSource бібліотек та інструментів розроблених мовою Python, що реалізують додаткові функції для роботи з текстом, HTTP запитами, мережею, зображеннями, інтерфейсом користувача. Однією з таких бібліотек є бібліотека *Grab*. Основними функціями цієї бібліотеки є підготовка мережевого запиту (cookies, http-заголовки), відправлення запитів, отримання відповідей сервера та їх попередня обробка, робота з DOM-деревом отриманої у відповідь сторінки [5]. Ми використали Grab для роботи з контентом Web сторінки, а саме для отримання текстової інформації із сторінки та очищенням її від HTML тегів.

Для реалізації Web-серверу ми скористалися найпоширенішим Python Framework для розробки Web-систем Django, дистрибутив якого містить також і Web-сервер [6].

Представлення отриманих результатів – важлива складова будь-якої системи видобування даних. У нашому випадку, отримані результати представлені у вигляді HTML сторінок.

## Результати тестування

Реалізована система показала хороші результати роботи.

Середня кількість понять, що є загальними і не являють собою приховані знання становить приблизно 30 – 40 % , а обсяг даних, що були прихованими до обробки запиту системою і були представлені у таблиці результатів відповідно становили 70 – 50 % з усіх отриманих даних в таблиці результатів.

Важливо зазначити, що якість роботи системи залежить від загальної кількості даних про об'єкт пошуку, що містяться у сховищі даних. Так при побудові профайлів користувачів, які є публічними особами кількість термінів, що викривають нові знання про об'єкт серед усіх отриманих результатів висока (понад 60 %), а для осіб, які не є публічними, а значить кількість інформації у сховищі даних про та-

ких осіб менша, результати дещо нижчі (40 – 50 %).

Як бачимо, система успішно здійснює пошук зв'язків і залежностей. Після здійснення користувачем додаткових запитів, у відповідь система надає нові зв'язки та нову інформацію про об'єкт пошуку.

## Висновки

Описаний алгоритм дозволяє здійснювати ефективний пошук зв'язків між об'єктом пошуку і даними Web-сторінок. Найсуттєвішою перевагою алгоритму є використання вже існуючого сховища даних. АПЗЗ використовує як сховище даних будь-яку існуючу ПС (або сховище даних), до яких існує доступ. Це дозволяє скоротити часові затрати і уникнути дублювання розробки уже існуючих систем. Другим привабливим фактором алгоритму є мовна незалежність. АПЗЗ дуже мало залежить від мови, що використовується користувачем. Навіть якщо не здійснювати обробку природної мови у процесі виконання алгоритму, результати роботи практично не змінюються. Часто обробку природних мов здійснює сама пошукова система. На кінець маємо відмітити і простоту реалізації.

Недоліками алгоритму є націленість на роботу з Web-сторінками. Деякою мірою алгоритм залежить від розміщення даних у ресурсі.

Суттєвою є і залежність від роботи пошукових систем. Алгоритм залежить як від якості пошуку, так і від швидкодії. Низька релевантність пошуку спричинить низьку якість результатів. Багато пошукових систем не дозволяють здійснювати велику кількість автоматичних запитів, що, за дуже складних аналізів 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 АПЗЗ. 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.

Шляхами удосконалення алгоритму бачиться покращення роботи з соціальними мережами і побудова дерева залежностей на кожному кроці взаємодії із ПС. Побудова деякого дерева, на кожному рівні якого зберігатимуться нові знайдені зв'язки допоможе відповісти на питання: як саме, або за рахунок чого об'єкт має зв'язок з іншим.

Програмна реалізація алгоритму показала хороші результати роботи. Окрім посилань на сторінки користувача, який є об'єктом запиту, у соціальних мережах користувач системи отримує таблицю із зв'язками (профайл) об'єкта пошуку з іншими об'єктами. Середнє відношення корисних даних у результативній таблиці складає 55 – 65 %.


1. *Глибовець М.М., Глибовець А.М., Поляков М.В.* Інтелектуальні мережі // Навчальний посібник, Дніпропетровськ, Нова ідеологія, 2014. – 464 с.
2. *Глибовець М.М., Жигмановський А.А., Заболотний Р.І., Захоженко П.О.* Веб сервіси оброблення документів // Національний університет "Києво-Могилянська академія". – К.: НаУКМА, 2012. – 212 с.
3. *Глибовец А.Н., Глибовец Н.Н., Покопцев Д.Е., Сидоренко М.О.* Структурированные данные и семантическая паутина: технологии Wiki // Проблеми програмування. – 2013. – № 1. – С. 45–67.
4. *Петренко А.І.* Grid і інтелектуальна обробка даних [Електронний ресурс]. – Режим доступу: http://datamining.netallted.cad.kiev.ua/downloads/DataMining.pdf
5. *grab* 0.6.29 : Python Package Index [Електронний ресурс] – Режим доступу: https://pypi.python.org/pypi/grab/0.6.29 – 2015 р.
6. *Django*: The Web framework for perfectionists with deadlines [Елекронний ресурс] – Режим доступу: https://www.djangoproject.com/

## References


1. *Glybovets M.M., Glybovets A.M., Poliakov M.V.* Intellectual networks, Dnipropetrovsk, newideology. – 2014. – 464 p.
2. *Glybovets M.M., Jigmanovskiy A.A., Zabolotniy R.I., Zahojenko P.O.* Webservices for documents processing. – K.: National university of "Kyiv-Mohyla academy". 2012. – 212 p.
3. *Glybovets M.M Glybovets A.M., Pokoptsev D.E., Sidorenko M.O.* Structured data and thesemantic web // Problems of programming. – 2013. – N 1. – P. 45–67.
4. *Petrenko A.I.* Grid and intelligent data processing [online] Available from: http://netallted.cad.kiev.ua/downloads/DataMining.pdf. [accesed: 2008]. – 2008.
5. *grab* 0.6.29 Python Package Index. [online] Available from: https://pypi.python.org/pypi/grab/0.6.29. [accesed © 1990–2015]. – 2015.
6. *Django* The Web framework for perfectionists with deadlines. [online] Available from: https://www.djangoproject.com/. [accesed: © 2005–2015]. – 2015.


Одержано 16.12.2015

***Про автора:***

*Глибовець Андрій Миколайович,*
кандидат фізико-математичних наук,
доцент кафедри мережних технологій
Кількість наукових публікацій в українських виданнях – 28.
Індекс Гірша – 3.
http://orcid.org/0000-0003-4282-481X

***Місце роботи автора:***

Національний університет
«Києво-Могилянська академія»,
04655, Київ, вул. Г. Сковороди 2.
Тел.: (044) 463 6985.
E-mail: andriy@glybovets.com.ua