УДК 004.021; 004.043; 004.62
UDC 004.021; 004.043; 004.62

ОПТИМІЗАЦІЯ ПОШУКУ ТЕКСТОВИХ ДАНИХ ЗА РАХУНОК ПОПЕРЕДНЬОЇ ОБРОБКИ ВХІДНИХ ДАНИХ

Гавриленко О.В., кандидат фізико-математичних наук, НТУУ «КПІ ім. Ігоря Сікорського», Київ, Україна, iem.gavrilenko@meta.ua, orcid.org/0000-0003-0413-6274,
Гетьманенко О.В., НТУУ «КПІ ім. Ігоря Сікорського», Київ, Україна, sasha.getmanenko@gmail.com, orcid.org/0000-0003-3445-3406

OPTIMIZATION OF SEARCHING PROCESS OF TEXT DATA DUE TO PRELIMINARY PROCESSING OF INPUT DATA

Gavrilenko O., Candidate of physical and mathematical Sciences, NTUU «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, iem.gavrilenko@meta.ua, orcid.org/0000-0003-0413-6274
Getmanenko O., NTUU «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine, sasha.getmanenko@gmail.com, orcid.org/0000-0003-3445-3406

ОПТИМИЗАЦИЯ ПОИСКА ТЕКСТОВЫХ ДАННЫХ ЗА СЧЕТ ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ВХОДНЫХ ДАННЫХ

Гавриленко Е.В., кандидат физико-математических наук, НТУУ «КПИ им. Игоря Сикорского», Киев, Украина, iem.gavrilenko@meta.ua, orcid.org/0000-0003-0413-6274
Гетьманенко А.В., НТУУ «КПИ им. Игоря Сикорского», Киев, Украина, sasha.getmanenko@gmail.com, orcid.org/0000-0003-3445-3406

Вступ. Важливою складовою сучасних технологій, бізнес-процесів є і залишається автоматизація. Багато існуючих процесів виробництва максимально переходять на автоматизовані системи з мінімальним втручанням людської діяльності, що легко пояснюється високими показниками ефективності, а також дозволяє зменшити відсоток помилок і уникнути так званого людського фактору.

Якщо говорити про сучасне програмне забезпечення, то його невід'ємною частиною є синхронізація, яка дозволяє переносити дані з одного додатку на інший без жодного втручання. Прикладом цього можуть бути системи Android, IOS, де дані переносяться на хмарне середовище і користувач, відповідно, може мати доступ до даних ресурсів із будь-якого свого девайсу. Аналогічна ситуація із web-розробками. Часто для авторизації пропонується обрати соціальну мережу, у якій Ви зареєстровані, і після успішної авторизації у обраній системі відбувається автоматична синхронізація (користувачу не потрібно повторно заповнювати профіль). Усе це набуло досить великої популярності. Для зручності сучасні програми містять API, яке дозволяє при наявності відповідних прав отримати чи зробити певну дію на ресурсі (це може бути отримання зображень, повідомлень). Популярні соціальні мережі містять такі засоби, завдяки яким є можливість залишати коментарі на різних ресурсах, ставити лайки, робити репости. Однак що робити, коли система не надає такої можливості або його власники припинили оновлення ресурсу чи вважають недоцільним витрачати кошти у даному напрямку, проте необхідно отримати дані? Найпопулярнішим і найпоширенішим рішенням залишається парсинг – технологія, яка дозволяє зчитувати і розпізнавати необхідні текстові дані заданого формату [1].

Парсинг сайтів є ефективним рішенням для автоматизації збору і зміни інформації. У порівнянні з людиною, комп'ютерна програма-парсер швидко обійде тисячі веб-сторінок; акуратно відокремить технічну інформацію від «людської»; безпомилково відбере потрібне і відкине зайве; ефективно упакує кінцеві дані в необхідному вигляді.

Результат (будь то база даних або електронні таблиці), звичайно ж, потребує подальшої обробки. Втім, подальші маніпуляції із зібраною інформацією вже до теми парсингу не належать.

Постановка проблеми: Дослідження засобів і методів парсингу є неоднозначною задачею. Опис і проведення експериментів з існуючими бібліотеками наведено у роботі [2]. У даній статті наводиться порівняння готових методів і бібліотек, проводиться аналіз часових витрат, витрат обсягу

оперативної пам'яті для кожного методу. Отож, для написання і створення парсингу під конкретну задачу, задля збереження швидкодії доцільніше написати свій регулярний вираз.

Застосування регулярних виразів має досить широку сферу, оскільки їх використання надає значно більше можливостей, ніж стандартний текстовий пошук. Якщо розглянути їх із середини, то вони являють собою кінцевий автомат, про особливості якого можна прочитати у роботі [3].

Існують так звані парсер-комбінатори, які являють собою функції, написані у стилі функціонального програмування. Їхня особливість полягає у початковому пошуку загального, а вже потім з кожним наступним кроком іде наближення до конкретного.

Якщо заданих швидкодій з використанням усього стеку нових технологій не вистачатиме, можна запропонувати наступні варіанти вирішення даної проблеми: збільшити потужності обчислювальної техніки, змінити бізнес-логіку чи систему.

Одне із вирішень даної проблеми – це написання коректного патерну для регулярного виразу. Писати варто з використанням стандартної документації, уникати непотрібних порівнянь і надлишкових записів. Для підтвердження ефективності даного підходу проведемо дослідження: напишемо три регулярні вирази для пошуку зображень на web-сторінках (будемо брати посилання з масиву розмірністю 65 елементів, який містять популярні сайти на українському Інтернет-просторі).

Таблиця 1 – Перелік регулярних виразів для дослідження
Table 1 – List of regular expressions for research

№	Патерн регулярного виразу (*синтаксис PHP*)
1 (11)	'/(img | src) = ("|')[^"\'>]+/i'
2 (21)	'~<img[^>]*(?<!_mce_)src\s?=\s?(["\'])((?^(?!\1).)*)[^>]*>~i'
3 (31)	'/<img(.*)src(.*)=(.*)"(.*)"[^"]+/U'

В результаті розрахунків отримано наступний графік, що свідчить про важливості вибору і попереднього дослідження складеного патерну.

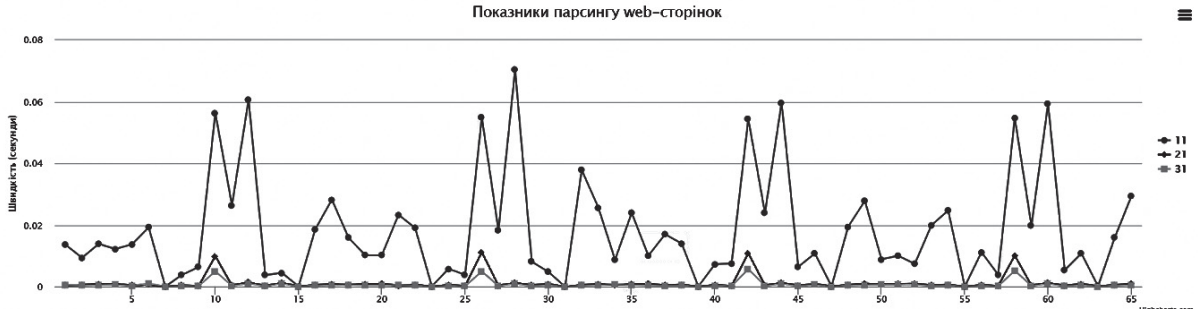

Рисунок 1 – Швидкість парсингу сторінки кожним із парсерів
Figure 1 – Parsing page speed for each parser

Поточний алгоритм (***алгоритм 1***) виглядає наступним чином:

1. *Зчитуємо сторінку.*
2. *Передаємо контент сторінки (текстові дані) на парсинг.*
   2.1. Шукаємо зображення.
   2.2. Перевіряємо коректність знайденого посилання (зображення).
   2.3. Додаємо зображення до масиву.
3. *Повертаємо список отриманих зображень.*

Наступним кроком для поліпшення швидкості виконання алгоритму є попередній аналіз сторінки. Потрібно обмежити обсяг вхідних даних, оскільки швидкість пропорційно залежить від розміру аналізованої інформації. Усім відома загальноприйнята структура сторінки:

```
<html>
  <head>
    <meta ...>
  <titile>Example web page</title>
  </head>
  <body>
    <h1>Header</h1>
   <p>First article</p>
   <p>Second article</p>
   <p>Third article</p>
  <img src="..." alt="image 1">
  <img src="..." alt="image 2">
  <img src="..." alt="image 3">
  </body>
</html>
```

Рисунок 2 – Загальна структура html-сторінки
Figure 2 – The general structure of the html page

У верхній частині (***head***) розташована інформація, яка містить технічну інформацію і яка не потрібна для аналізу. Після модифікації схема початкового алгоритму набуде наступного вигляду (***алгоритм 2***).

1. *Зчитуємо сторінку.*
2. *Відкидаємо контент header-y, footer-y.*
3. *Передаємо отриманий контент сторінки (body) на парсинг.*
   3.1. Шукаємо зображення.
   3.2. Перевіряємо коректність знайденого посилання (зображення).
   3.3. Додаємо зображення до масиву.
4. *Повертаємо список отриманих зображень.*

Даний алгоритм містить модифікацію лише вхідних даних. Дуже вузьким місцем у алгоритмі 2 є об'єм даних, який подається на аналіз. Для прискорення алгоритму потрібно правильно робити попередню обробку вхідної інформації, у даному випадку це *html* сторінка.

Окрім того, *<script>* є одним із надлишкових даних. Він жодним чином не вплине на отримані дані, проте може займати значний обсяг. Тому слід його відкинути ***(алгоритм 3)***.

1. *Зчитуємо сторінку.*
2. *Відкидаємо контент header-y, footer-y.*
3. *Відкидаємо контент script-ів з body.*
4. *Передаємо отриманий контент сторінки (body) на парсинг.*
   4.1. Шукаємо зображення.
   4.2. Перевіряємо коректність знайденого посилання (зображення).
   4.3. Додаємо зображення до масиву.
5. *Повертаємо список отриманих зображень.*

Перейдемо до проведення експериментальної частини: проведемо 65 досліджень на попередніх даних. Результати проведення експериментів наведено на рисунку 3.

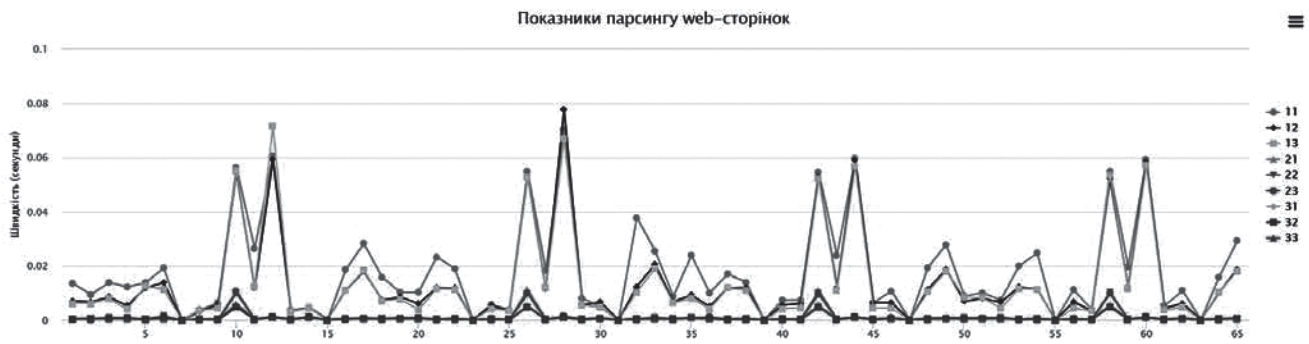

Рисунок 3 – Порівняння роботи парсеру з різним обсягом вхідних даних
Figure 3 – Comparison of the work of the parser with different inputs

Якщо проаналізувати даний графік, то ми бачимо, що швидкість безпосередньо залежить від розмірів вхідних даних, що і прагнули довести.

Для зручності приведемо дані кожного із патерну з використанням модифікованих алгоритмів окремо.

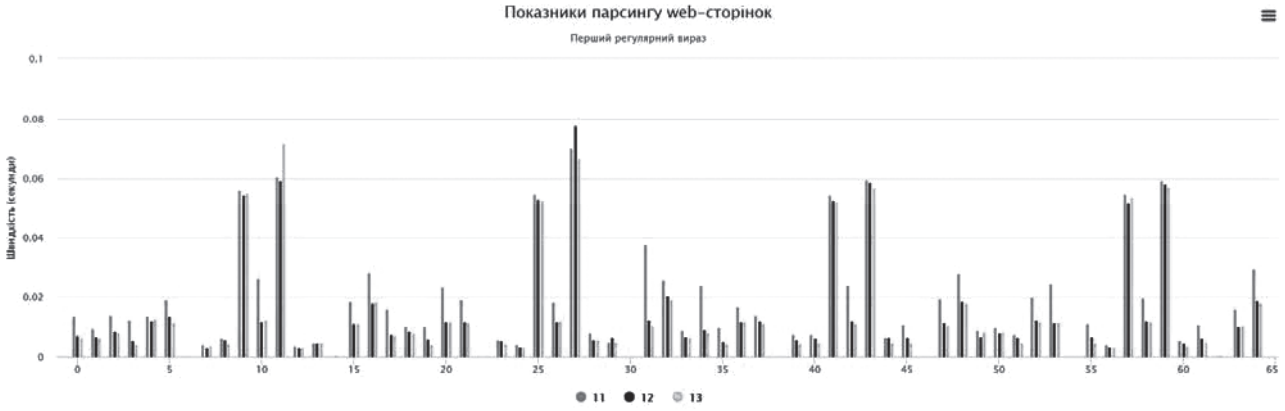

Рисунок 4 – Показники тривалості виконання першого парсеру
Figure 4 – Indicators of the duration of the first parser

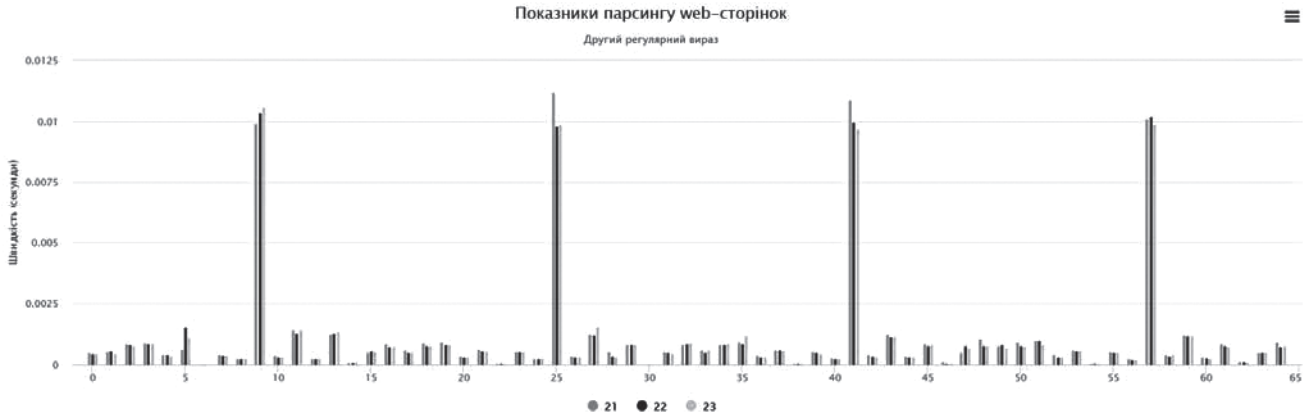

Рисунок 5 – Показники тривалості виконання другого парсеру
Figure 5 – Indicators of the duration of the second parser

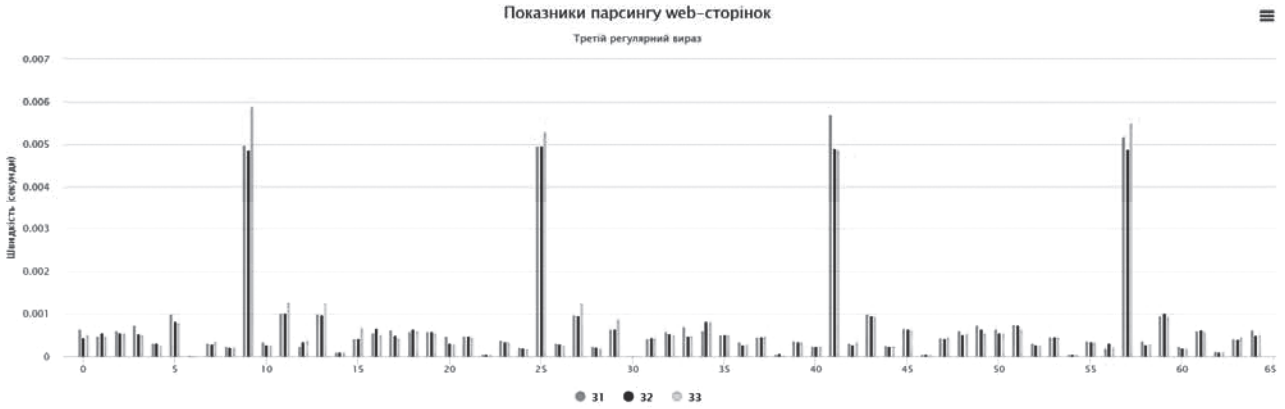

Рисунок 6 – Показники тривалості виконання третього парсеру
Figure 6 – Indicators for the duration of the third parser

Висновки. Аналіз графіків підтверджує обґрунтованість нашого вибору модифікації алгоритму за рахунок попередньої обробки вхідних даних. Проте інколи можуть спостерігатися розбіжності в результатах, у випадках, коли час без оптимізації кращий. Це пояснюється витратами часу на пошук і вилучення ***<head>*** , ***<script>***, ***<footer>***, обсяг яких може бути незначним.

Підведемо загальні підсумки: знайдемо середній час виконання кожного із алгоритмів для заданих патернів. Дані наведені у таблиці.

Таблиця 2 – Оцінка ефективності модифікованих алгоритмів
Table 2 – Estimation of the efficiency of modified algorithms

№	*Алгоритм 1*	*Алгоритм 2*	*Алгоритм 3*	*Ефективність*
1	0.0180420288672814	0.014000496497521034	0.013451510209303635	25,44%
2	0.001198867651132437	0.0011477066920353815	0.0011411520150991586	4,8%
3	0.0007529478806715745	0.00071257811326247	0.0007530102362999549	5,36%

Отож, можна зробити наступний висновок: оптимізація, яка була коротко описана і наведена у даній статті, дозволяє значно скоротити ресурси і прискорити виконання скрипту. Однак, варто пам’ятати про випадки, коли час без оптимізації буде кращим, і враховувати їх.

ПЕРЕЛІК ПОСИЛАНЬ

1. A.V. Getmanenko. What Graber and a Parser? / A.V. Getmanenko: Innovations in science and technology, 2016. – c. 35–36.

2. O.V. Gavrilenko. Search for the necessary content from unstructured data by example of finding images on the website. / O.V. Gavrilenko, A.V. Getmanenko, Informatics and Computer Science-ICS-2017. – 2017.

3. A.V. Getmanenko. Creating a parser-combinators with further optimization / A.V. Getmanenko: The theory and practice of modern science (c. Dnipro, 24-25 February 2017 y.). – Herson: Publishing house "Helvetyka" - 2017.

REFERENCES

1. A. Getmanenko. (2016). What Graber and a Parser? [Innovations in science and technology]. [in English].

2. O. Gavrilenko and A. Getmanenko. (2017). Search for the necessary content from unstructured data by example of finding images on the website. [Informatics and Computer Science-ICS-2017]. [in English].

3. A. Getmanenko. (2017). Creating a parser-combinators with further optimization. [The theory and practice of modern science (c. Dnipro, 24-25 February 2017 y.). – Herson: Publishing house "Helvetyka"]. [in English].

РЕФЕРАТ

Гавриленко О.В. Оптимізація пошуку текстових даних за рахунок попередньої обробки вхідних даних / О.В. Гавриленко, О.В. Гетьманенко // Вісник Національного транспортного університету. Серія «Технічні науки». Науково-технічний збірник. – К. : НТУ, 2018. – Вип. 1 (40).

Важливою складовою сучасних технологій, бізнес-процесів є і залишається автоматизація. Багато існуючих процесів виробництва максимально переходять на автоматизовані системи з мінімальним втручанням людської діяльності, що легко пояснюється високими показниками ефективності, а також дозволяє зменшити відсоток помилок і уникнути так званого людського фактору.

Якщо говорити про сучасне програмне забезпечення, то його невід’ємною частиною є синхронізація, яка дозволяє переносити дані з одного додатку на інший без жодного втручання. Прикладом цього можуть бути системи Android, IOS, де дані переносяться на хмарне середовище і користувач, відповідно, може мати доступ до даних ресурсів із будь-якого свого девайсу. Аналогічна ситуація із web-розробками. Часто для авторизації пропонується обрати соціальну мережу, у якій Ви зареєстровані, і після успішної авторизації у обраній системі відбувається автоматична синхронізація (користувачу не потрібно повторно заповнювати профіль). Усе це набуло досить великої популярності. Для зручності сучасні програми містять API, яке дозволяє при наявності відповідних

прав отримати чи зробити певну дію на ресурсі (це може бути отримання зображень, повідомлень). Популярні соціальні мережі містять такі засоби, завдяки яким є можливість залишати коментарі на різних ресурсах, ставити лайки, робити репости. Однак що робити, коли система не надає такої можливості або його власники припинили оновлення ресурсу чи вважають недоцільним витрачати кошти у даному напрямку, проте необхідно отримати дані? Найпопулярнішим і найпоширенішим рішенням залишається парсинг – технологія, яка дозволяє зчитувати і розпізнавати необхідні текстові дані заданого формату [1].

Парсинг сайтів є ефективним рішенням для автоматизації збору і зміни інформації. У порівнянні з людиною, комп'ютерна програма-парсер швидко обійде тисячі веб-сторінок; акуратно відокремить технічну інформацію від «людської»; безпомилково відбере потрібне і відкине зайве; ефективно упакує кінцеві дані в необхідному вигляді.

Результат (будь то база даних або електронні таблиці), звичайно ж, потребує подальшої обробки. Втім, подальші маніпуляції із зібраною інформацією вже до теми парсингу не належать.

Робота присвячена дослідженню засобів пошуку інформації заданого формату на неструктурованих даних. У якості рекомендованого методу пропонується створення парсеру побудованого на регулярних виразах. Наводиться схема простого алгоритму і описується подальша модифікація для покращення результатів швидкодії.

Об’єктами дослідження є популярні web-сторінки, інформацію яких було використано для проведення досліджень.

Мета даної роботи полягає у скороченні часу пошуку даних, а також зменшення обсягу використання оперативної пам’яті за рахунок доцільної побудови алгоритму, використовуючи найбільш ефективні методи попередньої обробки тексту.

Наведений алгоритм дозволяє скоротити час пошуку даних без жодних збільшень потужностей обчислювальної машини. Показники ефективності свідчать про доцільність застосування даного методу при пошуку на web-сторінках.

КЛЮЧОВІ СЛОВА: АНАЛІЗ ДАНИХ, АНАЛІЗ КОНТЕНТУ, ПАРСИНГ, НЕСТРУКТУРОВАНІ ДАНІ, ПАТЕРН, РЕГУЛЯРНІ ВИРАЗИ, МОДИФІКАЦІЯ АЛГОРИТМУ

ABSTRACT

Gavrilenko O.V., Getmanenko O.V. Optimization of searching process of text data due to preliminary processing of input dat. Visnyk National Transport University. Series «Technical sciences». Scientific and Technical Collection. – Kyiv: National Transport University, 2018. – Issue 1 (40).

Automation is an important component of modern technologies and business processes. Many existing production processes go as far as automated systems with minimal human intervention, easily explained by high efficiency indicators, and also allows to reduce the percentage of errors and avoid the so-called human factor.

If we talk about modern software, then its integral part is synchronization, which allows you to transfer data from one application to another without any intervention. An example of this can be Android, iOS, where the data is transferred to the cloud and the user, respectively, can access these resources from any of his devices. A similar situation with web-development. Often for authorization it is suggested to choose a social network in which you are registered, and after successful authorization, the system automatically synchronizes (the user does not need to re-fill in the profile). All this has become quite popular. For convenience, modern programs contain an API that allows you, if you have the appropriate rights, to get or make a certain action on the resource (this can be obtaining images, messages). Popular social networks contain such tools, thanks to which it is possible to leave comments on various resources, to put likes, to make reposts. However, what to do when the system does not provide such an opportunity or its owners have stopped updating the resource consider it inappropriate to spend money in this direction, but you need to get the data? The most popular and widespread solution is parsing - a technology that allows you to read and recognize the necessary text data of a given format [1].

Parsing sites is an effective solution for automating the collection and modification of information. In comparison with a human, a computer parser program quickly bypasses thousands of web pages, neatly separates technical information from "human"; unerringly select the right and discard the superfluous; effectively wraps the final data in the required form.

The result (be it a database or spreadsheets), of course, requires further processing. However, further manipulations with the collected information already to the topic of parsing do not belong.

The work is devoted to the research of information retrieval tools of a given format on unstructured data. As a recommended method, it is proposed to create a parser built on regular expressions. A scheme of a simple algorithm is given and further modification is described to improve the performance results.

The objects of research are popular web pages, the information of which was used for research.

The purpose of this work is to reduce the time of data retrieval, as well as reduce the amount of RAM usage due to the expedient construction of the algorithm, using the most effective methods of preliminary text processing.

The given algorithm allows to reduce the time of data search without adding the powers of the computer. Performance indicators indicate the appropriateness of using this method when searching on web pages.

KEYWORDS: DATA ANALYSIS, CONTENT ANALYSIS, PARSING, UNSTRUCTURED DATA, PATTERNS, REGULAR EXPRESSIONS, MODIFICATION OF ALGORITHM

РЕФЕРАТ

Гавриленко Е.В. Оптимизация поиска текстовых данных за счет предварительной обработки входных данных / Е.В. Гавриленко, А.В. Гетьманенко // Вестник Национального транспортного университета. Серия «Технические науки». Научно-технический сборник. – К.: НТУ, 2018. – Вып. 1 (40).

Важной составляющей современных технологий, бизнес-процессов является и остается автоматизация. Многие существующих процессов производства максимально переходят на автоматизированные системы с минимальным вмешательством человеческой деятельности, легко объясняется высокими показателями эффективности, а также позволяет уменьшить процент ошибок и избежать так называемого человеческого фактора.

Если говорить о современное программное обеспечение, то его неотъемлемой частью является синхронизация, которая позволяет переносить данные с одного приложения на другое без всякого вмешательства. Примером этого могут быть системы Android, IOS, где данные переносятся на облачную среду и пользователь, соответственно, может иметь доступ к данным ресурсам с любого своего девайса. Аналогичная ситуация с web-разработками. Часто для авторизации предлагается выбрать социальную сеть, в которой Вы зарегистрированы, и после успешной авторизации в выбранной системе происходит автоматическая синхронизация (пользователю не нужно повторно заполнять профиль). Все это получило достаточно большую популярность. Для удобства современные программы содержат API, которое позволяет при наличии соответствующих прав получить или сделать определенное действие на ресурсе (это может быть получение изображений, сообщений). Популярные социальные сети содержат такие средства, благодаря которым есть возможность оставлять комментарии на различных ресурсах, ставить лайки, делать репосты. Однако что делать, когда система не предоставляет такой возможности или его владельцы прекратили обновление ресурса считают нецелесообразным тратить средства в данном направлении, однако необходимо получить данные? Самым популярным и распространенным решением остается парсинг - технология, которая позволяет считывать и распознавать необходимые текстовые данные заданного формата [1].

Парсинг сайтов является эффективным решением для автоматизации сбора и изменения информации. По сравнению с человеком, компьютерная программа-парсер быстро обойдет тысячи веб-страниц, аккуратно отделит техническую информацию от «человеческой»; безошибочно отберет нужное и отбросит лишнее; эффективно упакует конечные данные в требуемом виде.

Результат (будь то база данных или электронные таблицы), конечно же, требует дальнейшей обработки. Впрочем, дальнейшие манипуляции с собранной информацией уже к теме парсинга не принадлежат.

Работа посвящена исследованию средств поиска информации заданного формата на неструктурированных данных. В качестве рекомендованного метода предлагается создание парсера построенного на регулярных выражениях. Приводится схема простого алгоритма и описывается дальнейшая модификация для улучшения результатов быстродействия.

Объектами исследования являются популярные web-страницы, информацию которых были использованы для проведения исследований.

Цель данной работы заключается в сокращении времени поиска данных, а также уменьшение объема использования оперативной памяти за счет целесообразной построения алгоритма, используя наиболее эффективные методы предварительной обработки текста.

Приведенный алгоритм позволяет сократить время поиска данных без прибавок мощностей вычислительной машины. Показатели эффективности свидетельствуют о целесообразности применения данного метода при поиске на web-страницах.

КЛЮЧЕВЫЕ СЛОВА: АНАЛИЗ ДАННЫХ, АНАЛИЗ КОНТЕНТА, ПАРСИНГ, НЕСТРУКТУРИРОВАННЫЕ ДАННЫЕ, ПАТТЕРН, РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ, МОДИФИКАЦИИ АЛГОРИТМА

АВТОРИ:

Гавриленко Олена Валеріївна, кандидат фізико-математичних наук, доцент, доцент кафедри АСОІУ, ФІОТ, НТУУ «КПІ ім. Ігоря Сікорського», https://orcid.org/0000-0003-0413-6274, e-mail: iem.gavrilenko@meta.ua, +380935768058, Україна, Київ, вул. Політехнічна, 41, 18 корпус, к. 430;

Гетьманенко Олександр Володимирович, студент 2-го курсу магістратури, АСОІУ, ФІОТ, НТУУ «КПІ ім. Ігоря Сікорського», https://orcid.org/0000-0003-3445-3406, e-mail: sasha.getmanenko@gmail.com, +380967494346, Україна, Київ, вул. Політехнічна, 41, 18 корпус.

AUTHORS:

Gavrilenko Olena, Candidate of physical and mathematical Sciences, Associate Professor, Associate Professor of Department AIPSM, FICT, e-mail: iem.gavrilenko@meta.ua, +380935768058, Ukraine, Kyiv, Polytehnycheskaya st., 41, 18 buld., r. 430;

Getmanenko Oleksandr, master's course student of AIPSM, FICT, NTUU «Igor Sikorsky Kyiv Polytechnic Institute», https://orcid.org/0000-0003-3445-3406, e-mail: sasha.getmanenko@gmail.com, +380967494346, Ukraine, Kyiv, Polytehnycheskaya st., 41, 18 buld.

АВТОРЫ:

Гавриленко Елена Валериевна, кандидат физико-математических наук, доцент, доцент кафедры АСОИУ, ФИВТ, НТУУ «КПИ им. Игоря Сикорского», https://orcid.org/0000-0003-0413-6274, e-mail: iem.gavrilenko@meta.ua, +380935768058, Украина, Киев, ул. Политехническая, 41, 18 корпус, к. 430;

Гетьманенко Александр Владимирович, студент 2-го курса магистратуры, АСОИУ, ФИВТ, НТУУ «КПИ им. Игоря Сикорского», https://orcid.org/0000-0003-3445-3406, e-mail: sasha.getmanenko@gmail.com, +380967494346, Украина, Киев, ул. Политехническая, 41, 18 корпус.

РЕЦЕНЗЕНТИ:

Прокудін Г.С., завідувач кафедри міжнародних перевезень та митного контролю Національного транспортного університету, доктор технічних наук, професор, Київ, Україна

Кирилюк В.С., провідний науковий співробітник Інституту механіки ім. С.П. Тимошенка НАНУ, доктор фізико-математичних наук, старший науковий співробітник, Київ, Україна

REVIEWER:

Prokudin G.S., head of international transportation and customs control of National Transport University, Doctor Sciences of Engineering, Professor, Kyiv, Ukraine

Kyrylyuk V.S., Leading Researcher of the Institute of Mechanics SP Tymoshenko National Academy of Sciences, Doctor Sciences of Physical and Mathematical, Senior Researcher, Kyiv, Ukraine