УДК 004.9
DOI: 10.25559/SITITO.15.201902.413-420

Разработка приложения веб-скрапинга с возможностями обхода блокировок

А. А. Москаленко*, О. Р. Лапонина, В. А. Сухомлин
Московский государственный университет имени М.В. Ломоносова, г. Москва, Россия
119991, Россия, г. Москва, ГСП-1, Ленинские горы, д. 1
* alekmosk25@gmail.com

Аннотация

Рассмотрен веб-скрапинг – процесс извлечения данных со страниц веб-сайтов в интернете с помощью автоматизации обращений к веб-сайту. С развитием интернета важность веб-скрапинга возросла, и более половины интернет-трафика на веб-сайты (за исключением потокового, т.е. аудио и видео) создается автоматизированными средствами, так называемыми ботами.

Статья посвящена исследованию процесса веб-скраппинга и изучению проблемы блокировки веб-скраперов в сети Интернет. Рассматриваются основные принципы и понятия процесса веб-скрапинга. Проводится обзор существующих решений для веб-скрапинга, выделяются основные достоинства и недостатки веб-скраппинга с возможностью обхода блокировок. Рассматриваются причины блокировки веб-скраперов веб-сайтами, выделены признаки, по которым веб-сайты определяют и блокируют веб-скраперы. Исследуются приемы для обхода блокировок веб-скраперов и их влияние на процесс веб-скрапинга.

Предлагается программа, разработанная на языке программирования Python, которая использует приемы для обхода блокировок веб-скраперов. Программа имеет графический интерфейс, разработанный с помощью фреймворка Tkinter для создания политики веб-скраппинга. Для обхода блокировок веб-скраперов используется фреймворк с открытым исходным кодом для автоматизации действий пользователя в браузере Selenium WebDriver. Сравнительный анализ работы веб-скраперов показал, что использование созданных в работе модулей позволяет обойти блокировки веб-скрапинга.

Ключевые слова: веб-скрапинг, Selenium WebDriver, обход блокировок веб-скраперов, ротация IP-адресов, веб-приложение.

Для цитирования: *Москаленко А. А., Лапонина О. Р., Сухомлин В. А.* Разработка приложения веб-скрапинга с возможностями обхода блокировок // Современные информационные технологии и ИТ-образование. 2019. Т. 15, № 2. С. 413-420. DOI: 10.25559/SITITO.15.201902.413-420

Developing a Web Scraping Application with Bypass Blocking

A. A. Moskalenko*, O. R. Laponina, V. A. Sukhomlin
Lomonosov Moscow State University, Moscow, Russia
1, Leninskie gory, Moscow 119991, Russia
* alekmosk25@gmail.com

Abstract

Web-scraping is a process of extracting data from web-pages on the Internet by automating web-sites requests. Importance of web-scraping is increased with developing of the Internet. And more than half of Internet traffic (except for streaming, i.e. audio and video) is created by automated means, so-called bots.

The article is devoted to the study of the process of web-scraping and the problem of blocking web scrapers on the Internet. We consider the basic principles and concepts of web scraping process and classification of web scrapers. A review of existing web-scraping solutions is carried out, highlighting the main advantages and disadvantages of web-scraping bypassing locks. The reasons for blocking web scrapers by websites are considered, highlighting the signs by which websites determine and block web scrapers. We investigate techniques for bypassing web-scraper locks and their impact on the web-scraping process.

A program developed in the Python programming language that uses techniques to bypass web-scrapper locks is proposed. The program has a graphical interface developed using the Tkinter framework to create a web-scraping policy. Web scrapers bypassing blocking techniques use an open source framework to automate user actions in the Selenium WebDriver browser. A comparative analysis of the work of web scrapers showed that the use of the modules created in the work allows you to bypass the blocking of web scraping.

Keywords: Web-scraping, Selenium WebDriver, bypass web scraper locks, IP address rotation, web-application.

For citation: Moskalenko A.A., Laponina O.R., Sukhomlin V.A. Developing a Web Scraping Application with Bypass Blocking. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2019; 15(2):413-420. DOI: 10.25559/SITITO.15.201902.413-420

Введение

Веб-скрапинг (web-scrapping) – процесс извлечения данных со страниц веб-сайтов в интернете с помощью автоматизации обращений к веб-сайту. Копирование списка контактов из веб-каталога является примером скрапинга. Для сбора больших объемов данных необходима автоматизация, и веб-скраперы выполняют именно эту функцию. С юридической точки зрения процесс веб-скрапинга не может считаться незаконным, поскольку извлекается информация, доступная всем[1].

С развитием интернета важность веб-скрапинга возросла, и по данным исследования компании в области аудита информационной безопасности[2] [1] более половины (52%) интернет-трафика на веб-сайты (за исключением потокового, т.е. аудио и видео) создается автоматизированными средствами, так называемыми ботами.

Существует два основных типа веб-скраперов: веб-пауки (web scrappers) и веб-краулеры (web-crawlers, поисковые роботы). Основной целью обоих типов является извлечение данных из веб-сайтов, но между ними есть определенные различия.

1. Веб-паук построен специально для обработки структуры конкретного веб-сайта. Парсер использует структуру сайта для извлечения отдельных элементов данных[3]. Элементами данных могут быть имена, телефоны, цены, изображения и т. д. Примером являются службы SERP (Search Engine Results Page) monitoring services.
2. Веб-крауллинг в основном относится к загрузке и хранению содержимого большого количества веб-сайтов, быстрому поиску и обновлению базы данных. Примером веб-краулера являются роботы поисковых систем.

Эффективность поисковой системы во многом определяется свойствами используемых поисковых роботов. Googlebot – это пример поискового робота. Робот Googlebot обходит интернет по ссылкам с одной страницы на другую. Компания Google использует эту информацию для извлечения всех видов данных, чтобы сделать поиск наиболее эффективным. Все остальные поисковые системы, такие как Яндекс, Майл.Ру, используют свои боты аналогичным образом.

Веб-скрапинг автоматизирует следующие задачи:

Веб-скраперы используются для автоматизации следующих задач:

- Копирование данных из интернета
- Поиск необходимой информации
- Мониторинг обновлений информации на сайтах (например, купить товар, если его цена уменьшилась или этот товар появился в наличии)

Веб-скрапинг используется следующими прикладными системами:

- Поисковые системы – технологии поисковых систем предназначены для поиска данных в интернете.
- Агрегаторы контента – почти все агрегаторы контента используют веб-скрапинг. Например, новостные агрегаторы часто выделяют новые статьи из новостных веб-сайтов, чтобы предоставить самую актуальную информацию для своих пользователей[4].
- Научные и маркетинговые исследования. Для статистического анализа какой-либо сферы деятельности или рынка сбыта необходимо большое количество данных за длительные промежутки времени.
- Наборы данных для машинного обучения. Веб-скраперы преобразуют данные из интернета в структурированный набор данных для анализа.

Веб-скрапинг с точки зрения администратора сайта

Некоторые администраторы веб-сайтов противодействуют действиям веб-скраперов. Активность веб-скраперов может негативно сказаться на производительности веб-сервера. Кроме того, собранную информацию могут использовать для анализа конкурентов и получения преимущества на рынке.

Таким образом, важно исследовать процесс веб-скрапинга и с точки зрения защиты информации администраторами веб-сайтов, и с точки зрения веб-скрапера, которому требуется иметь возможность обойти блокировки.

Цели работы

Целью данной работы является разработка программы для веб-скрапинга с использованием приемов для обхода блокировок веб-скраперов. Для этого необходимо изучить процесс веб-скрапинга, технологии и приемы, которые используются для извлечения информации из веб-сайтов, способы блокировки веб-скраперов.

1. Изучение технологий веб-скрапинга для извлечения информации из веб-сайтов и способов блокировки веб-скраперов
2. Разработка программы веб-скрапинга с возможностью обхода блокировок веб-скраперов

Исследование процесса веб-скрапинга

Процесс веб-скрапинга состоит из трех основных этапов:

- Извлечение кода веб-страниц из интернета. На первом этапе программа совершает HTTP-запросы по URL-адресам в соответствии с логикой работы и извлекает HTML-код страниц.
- Извлечение необходимой информации из кода веб-страниц. На этом этапе с помощью специальных механизмов (регулярные выражения, HTML-парсеры, искусственный

[1] *Kalwar S.* Is scraping and crawling to collect data illegal? [Электронный ресурс]. Mar 31, 201. URL: https://www.quora.com/Is-scraping-and-crawling-to-collect-data-illegal дата обращения 01.06.19)

[2] *Zeifman I.* Bot Traffic Report 2016 [Электронный ресурс]. Jan 24, 2017. URL: https://www.imperva.com/blog/bot-traffic-report-2016/ (дата обращения 01.06.19)

[3] *Rezai A.* Web Crawling and Screen Scraping – the Legal Position. [Электронный ресурс]. 6th February 2017. URL: https://parissmith.co.uk/blog/web-crawling-screen-scraping-legal-position/ (дата обращения 01.06.19)

[4] *Joyce G.* Data Reveals the GRAMMYs 2017 Highlights on Social Media. [Электронный ресурс]. 13 February, 2017. URL: https://www.brandwatch.com/blog/react-grammys-2017-highlights/ (дата обращения 01.06.19)

интеллект) происходит выделение необходимой информации из HTML кода.
- Сохранение структурированных данных в таблицах или базах данных.

A. Технологии для извлечения данных
- Регулярные выражения – набор шаблонов, может использоваться для выполнения задач сопоставления строк шаблонам и обработки данных HTML. Этот метод полезен для обработки простых данных одного типа, таких как получение списка всех номеров телефонов с веб-страницы. Но не подходит для более сложных заданий, например, извлечения полей различных типов со страниц описания продукта на веб-сайте. Регулярные выражения являются полезным инструментом для преобразования и выделения необходимых данных.
- Веб-парсеры – специализированные библиотеки для обработки HTML-страниц. Часто веб-сайт содержит базовый шаблон для описания страницы и формирует ее контент из полей базы данных.
- Автоматический скрапинг с использованием искусственного интеллекта – этот метод является более сложным с точки зрения технологий и используется, когда необходимо извлекать данные из нескольких веб-сайтов, которые содержат похожую информацию. Веб-скраперы можно обучать с помощью моделей машинного обучения для извлечения данных из веб-страниц.

B. Определение веб-скрапинга веб-сайтами

Веб-сайты определяют веб-скраперы по характерным для веб-скраперов признакам:
1. Необычное количество трафика с одного IP-адреса (например, пользователь переходит на новую страницу сайта каждую секунду сотни раз)
2. Выполнение повторяющихся задач (основано на том, что пользователь не будет выполнять одни и те же задачи много раз)
3. Использование ссылок, которые не видны обычному пользователю, но содержатся в коде веб-страницы

C. Способы блокировки веб-скраперов
1. Запретить доступ на сайт с определенного IP-адреса.
2. Запретить идентификатор пользователя, с использованием которого веб-скрапер, являющийся с точки зрения администратора сайта злоумышленником, заходит на сайт. В этом способе доступ к сайту осуществляется только аутентифицированными пользователями.

Рекомендации для веб-скрапинга для преодоления обнаружения

1. Настроить скорость работы веб-скрапера

Настроить скорость работы веб-скрапера на оптимальную скорость обхода после нескольких пробных запусков и периодически проверять настройки веб-скрапера, так как сам сайт может меняться со временем. Поместить случайные программные вызовы между запросами, добавьте задержки после обхода некоторого количества страниц и регулировать количество одновременных запросов. Эти методы делают работу веб-скрапера похожей на работу человека.

2. Использовать прокси-серверы и ротацию IP-адресов

Если перед веб-скрапером стоит цель не вызывать подозрение, следует периодически менять IP-адрес. Прокси-серверы с ротацией IP-адресов циклически меняют IP-адреса. Таким образом, увеличивается вероятность равномерного использования всех доступных IP-адресов и снижается вероятность блокировки на сайте.

Веб-скрапер легко определить, если он регулярно отправляет похожие запросы с одного IP-адреса, если же использовать ротацию IP-адресов, то обнаружение становится сложнее. Существует несколько способов изменить IP-адрес. Могут помочь сервисы VPN, прокси-серверы, технологии TOR. Кроме того, различные коммерческие провайдеры также предоставляют услуги по автоматическому изменению IP адресов.

Идея ротации IP-адресов во время веб-скрапинга состоит в следующем: можно сделать так, чтобы повторяющиеся запросы отправлялись с различных IP-адресов. Таким образом, создается впечатление, что несколько реальных пользователей совершают запросы к серверу из различных мест.

3. Изменение пользовательского агента (User Agent)

Агент пользователя – это строка, которую отправляет веб-браузер для своей идентификации, агент пользователя содержит информацию о системе и браузере от которой приходит запрос: имя браузера, версию операционной системы и другие параметры. По агенту пользователя можно определить параметры системы: название операционной системы, версию и разрядность, разрешение экрана. Можно определить, какое устройство использует пользователь браузера, компьютер, телефон или планшет.

Каждый запрос, сделанный из веб-браузера, содержит заголовок агента пользователя, использование одного и того же агента пользователя приводит к обнаружению веб-скрапера. Способ обойти это обнаружение – подделывать агент пользователя и изменять его при каждом запросе на веб-сайт.

Если сайт заблокировал веб-скрапер, то можно попробовать обойти эту блокировку заменой агента пользователя или отправкой случайных записей в строке агента пользователя. Таким образом, у веб-сервера создается впечатление, что к нему приходят запросы от различных браузеров, а не из одного. Стоит отметить, что рандомизация строки агента пользователя без изменения IP-адресов почти бесполезна.

Обзор используемых инструментальных средств

A. MechanicalSoup и BeautifulSoup

MechanicalSoup и BeautifulSoup – библиотеки с открытым исходным кодом, написаны на языке Python. Основная задача библиотеки MechanicalSoup – имитация поведения человека в веб-браузере (прокручивание страницы, ввод данных, нажатие кнопок), извлечение кода веб-страницы и передача на обработку библиотеке BeautifulSoup.

Основная задача библиотеки BeautifulSoup – извлечение данных из HTML-кода. Библиотека использует CSS-селекторы для извлечения необходимой информации.

Преимущества
- Позволяет автоматизировать действия, которые выполняет человек в браузере (от перехода по ссылке, до нажа-

тия кнопок и ввода данных)
- Предоставляет возможность использовать различные библиотеки для обработки кода веб-страницы (lxml-parser, html-parser, собственные библиотеки)
- Недостатки
- Не поддерживает JavaScript (нет возможности обрабатывать объекты, которые генерирует JavaScript)
- Не поддерживает библиотеку XPath – язык запросов к элементам XML-документов)

B. Selenium WebDriver

Selenium WebDriver – набор библиотек с открытым исходным кодом для различных языков программирования. Эти библиотеки используются для отправки HTTP-запросов с помощью протокола JsonWireProtocol. В этих запросах указаны действия, которые должен совершить браузер в рамках текущей сессии. Примеры команд: переход по ссылкам, поиск и нажатие кнопок, извлечение и парсинг текста страницы.

Selenium WebDriver используется для решения сложных задач, он способен извлекать данные с сайтов, на которых контент формируется динамически (меняется со временем или при некоторых действия в браузере). Selenium использует реальный веб-браузер для доступа на сайт, таким образом активность программы ничем не отличается от активности реального пользователя в браузере. При загрузке страницы с помощью Selenium браузер загружает все веб-ресурсы и выполняет JavaScript на странице.

Архитектура **Selenium WebDriver**

Selenium WebDriver состоит из четырех компонентов

1. API Selenium для языка программирования. Представляет собой набор команд и объектов (интерфейс) для создания программ на основе Selenium.

2. WebDriver (HTTP Server). Каждый браузер содержит отдельный драйвер браузера – программного интерфейса для выполнения команд пользователя. Драйверы браузера взаимодействуют с соответствующим браузером, не раскрывая внутреннюю логику функциональности браузера. Когда драйвер браузера получает любую команду, эта команда будет выполнена в соответствующем браузере, и ответ вернется в виде ответа HTTP. Каждый драйвер браузера использует HTTP-сервер для получения HTTP-запросов. Как только URL-адрес достигнет драйвера браузера, драйвер браузера передаст этот запрос реальному браузеру по HTTP. Затем команды программы (логика работы веб-скрапинга) будут выполняться в браузере.

3. JSON Wire Protocol – протокол, который используется для передачи информации между сервером и клиентским приложением (пользователем). JavaScript Object Notation (JSON) используется для представления объектов со сложными структурами данных.

4. Браузер. Selenium поддерживает браузеры Chrome, Firefox, Opera и другие.

Преимущества
- Позволяет автоматизировать действия, которые выполняет человек в браузере (от перехода по ссылке, до нажатия кнопок и ввода данных)
- Поддерживает работу с объектами, которые динамически генерирует JavaScript
- Активность в браузере с использованием Selenium WebDriver ничем не отличается от действий человека в браузере
- Открытый исходный код и поддержка популярный языков программирования (Java, C#, Python)

Недостатки
- Для работы с браузером требуется большое количество ресурсов системы (оперативная память и загрузка процессора)
- Медленная скорость работы из-за ожидания полной загрузки страницы и выполнения JavaScript кода

Scrapy – специализированный фреймворк для веб-парсинга, написанный на Python. Предоставляет все необходимые инструменты от извлечения данных из интернета до сохранения данных в необходимом формате. Одним из главных преимуществ этого фреймворка является встроенная возможность асинхронной работы. Если необходимо обработать большое количество данных, то Scrapy имеет возможность разделить выполнение задачи по нескольким потокам. Scrapy содержит встроенные инструменты экспорта форматов, таких как JSON, XML и CSV.

Преимущества
- Высокая скорость работы
- Подходит для скрапинга большого количества данных

Недостатки
- Не поддерживает JavaScript
- Не универсальный (например, не поддерживает Python 3 под ОС Windows)

Описание программы

Анализ существующих решений показал, что самым подходящим инструментом веб-скрапинга, на основе которого будет разработана программа является Selenium WebDriver. Решение было принято по следующим причинам:

1. Selenium WebDriver использует полноценный браузер для своего функционирования. Это усложняет процесс определения веб-скрапера и делает веб-скрапинг нагляднее, так как активность программы видна в веб-браузере
2. Существует встроенная поддержка JavaScript и возможность расширять функциональность программы (выполнение скриптов в браузере)
3. Открытый исходный код и активное сообщество пользователей
4. Недостаток, связанный со скоростью работы не является существенным для поставленной задачи

A. Требования к программе:

1. Программа должна иметь графический интерфейс для сбора требований к процессу веб-скрапинга
2. Программа должна иметь следующие модули для обхода блокировок процесса веб-скрапинга:
   a. Модуль ротация IP-адресов
   b. Модуль для изменения пользовательского агента
   c. Модуль для включения случайных событий в работу программы
3. Модуль оценки работы программы

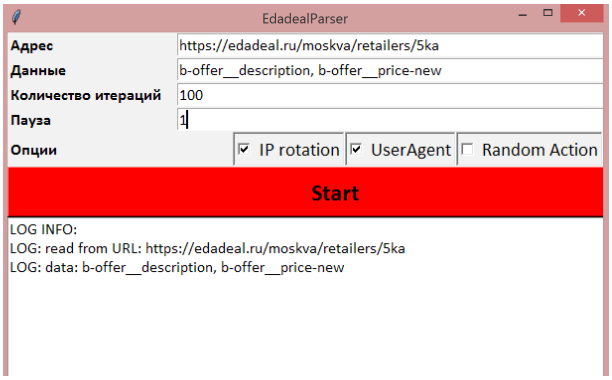
B. Интерфейс создания политики работы веб-скрапера

Для разработки интерфейса создания политики работы скрапера используется библиотека Tkinter. Интерфейс позволяет выбирать опции для процесса веб-скрапинга и содержит следующие поля:

- Адрес – ссылки на веб-страницы, которые будут обработаны программой (указываются через запятую)
- Данные – CSS-селекторы данных, которые необходимо извлечь (указываются через запятую)
- Выбор модулей, которые будут использоваться во время работы программы
- Количество итераций – сколько раз программа будет выполнять скрапинг
- Пауза – промежуток времени между итерациями работы программы (указан в секундах)

После ввода данных и нажатия кнопки “Start” программа последовательно совершает запросы к веб-сайту, извлекает HTML-код веб-страниц и преобразует его в структурированный набор данных.

Графический интерфейс содержит поле для вывода информации о текущем состоянии программы, сообщения об ошибках.



Р и с. 1. Графический интерфейс программы

F i g. 1. Program GUI

C. Модуль для изменения пользовательского агента

Модуль изменения пользовательского агента состоит из списка существующих пользовательских агентов, которые меняются при каждой итерации работы программы и кода для изменения пользовательского агента в браузере.

D. Модуль ротации IP-адресов

Модуль ротации IP-адресов состоит из двух частей – функционала для изменения IP-адресов и функционала для использования прокси-сервера.

Прокси-сервер – совокупность программ, выполняющих роль посредника между пользователем и целевым сервером, позволяющих клиентам выполнять запросы через вспомогательные серверы к другим сетевым службам и получать ответы на свои запросы. В данной работе прокси-сервер используется для анонимизации доступа к веб-сайту. Прокси сервер будет скрывать информацию о том, кто совершает запрос к веб-сайту. Веб-сайт получает информацию только о IP-адресе прокси-сервера.

Функционал для изменения IP-адресов изменяет адрес прокси-сервера после каждой итерации. В данной работе используются бесплатные прокси-сервера, адреса которых доступны на сайте https://free-proxy-list.net/anonymous-proxy.html.

E. Модуль для включения случайных событий в работу программы

Случайные события нужны для того, чтобы замаскировать однотипные действия программы под действия реального пользователя. Для создания случайных событий в работе используется класс ActionChains из библиотеки Selenium WebDriver. ActionChains используется для автоматизации взаимодействий с веб-сайтом.

Функции способны выполнять следующие действия:

- движение по страницы (аналогично движению колесика мыши)
- нажатие кнопок клавиатуры
- нажатие кнопок мыши
- перезагрузка страницы
- переход на следующую и предыдущую страницы

F. Модуль оценки работы процесса веб-скрапинга

Основная задача этого модуля – проверять правильность работы программы и оценка таких параметров, как скорость работы и количество ошибок или сбоев.

Модуль запускается в начале работы программы и проверяет действительно ли изменены IP-адрес, пользовательский агент. В течении всего времени работы программы модуль вычисляет время работы программы, время обработки одной страницы, обрабатывает ошибки, если они происходят.

Правильность изменения IP-адреса проверяется с помощью сайта whatismyipaddress.com, который автоматически определяет IP-адрес того, кто зашел на сайт. Программа отправляет запросы к этому веб-сайту и получает в ответ IP-адрес. Далее полученный IP-адрес проверяется. Аналогичным образом проверяется правильность изменения пользовательского агента с помощью веб-сайта https://httpbin.org/user-agent.

G. Схема работы программы

1. Заполнение формы сбора требований к тестированию веб-сайта
2. После нажатия кнопки «Start» запускается Selenium WebDriver
3. Программа начинает отправлять запросы к веб-серверу сайта с целью веб-скрапинга.
4. Сохраняет результаты работы в следующем формате:

<Номер итерации> <Время выполнения запроса> <Результат выполнения OK/ERROR> <Типы настройки [Empty, IP rotation, UserAgent, RandomAction]>

Результаты работы программы предназначены для анализа веб-сайта на возможность блокировать веб-скраперы и анализа влияния модулей для обхода блокировок на результаты веб-скрапинга.

Используя модули для обхода блокировок получилось выполнить веб-скрапинг сайта https://yandex.ru/sprav. Тестирование сайта показало, что модуль IP rotation помогает обходить блокировки на данном сайте. Сравнение работы веб-скраперов

Т а б л и ц а 1. **Сравнение работы веб-скраперов**

T a b l e 1. **Comparison of Web Scrapers**

Тип веб-скрапера	Номер итерации, с которой начинаются блокировки	Среднее время одной итерации (сек)
Без обхода блокировок	5	5.479
С обходом блокировок	-	6.862

Заключение

В данной работе исследуется процесс веб-скрапинга, спроектирована программа, которая выполняет все этапы процесса веб-скрапинга. Реализованы приемы для обхода блокировок

веб-скраперов. Модуль оценки работы программы проверяет правильность изменения IP-адресов и пользовательского агента. На основании модулей программы разработан веб-скрапер сайта ЯндексСправочник. Сравнительный анализ работы веб-скраперов показал, что использование разработанных модулей позволяет обойти блокировки веб-скрапинга и незначительно увеличивают время работы программы.

References

[1] Moeller R.R. IT Audit, Control, and Security. Hoboken: John Wile & Sons, Inc., 2010. 667 pp. (In Eng.)

[2] Gundecha U. Selenium Testing Tools Cookbook. Birmingham B3 2PB, UK.: Packt Publishing, 2015. (In Eng.)

[3] Mitchell R. Web Scraping with Python. USA.: O'Reilly Media, 2015. (In Eng.)

[4] Hajba G. Website Scraping with Python: Using BeautifulSoup. USA.: O'Reilly Media, 2018. (In Eng.)

[5] Nair G. Getting Started with Beautiful Soup. USA.: Packt Publishing, 2014. (In Eng.)

[6] Shrenk M. Webbots, spiders, and screen scrapers. USA.: Packt Publishing, 2012. (In Eng.)

[7] Buelta J. Python Automation Cookbook. USA.: Packt Publishing, 2018. (In Eng.)

[8] Koundal D. Ontology Based Crawler: Semantic web application USA.: Lambert, 2013. (In Eng.)

[9] Ferrara E., De Meo P., Fiumara G., Baumgartner R. Web Data Extraction, Applications and Techniques: A Survey. *Knowledge-Based Systems.* 2014; 70:301-323. (In Eng.) DOI: 10.1016/j.knosys.2014.07.007

[10] Liang H., Zhu J.J.H Big Data, Collection of (Social Media, Harvesting). *The International Encyclopedia of Communication Research Methods*. 2017; 1-18. (In Eng.) DOI: 10.1002/9781118901731.iecrm0015

[11] Hirschey J.K. Symbiotic Relationships: Pragmatic Acceptance of Data Scraping. *Berkeley Technology Law Journal.* 2014; 29:897-927. Available at: https://www.jstor.org/stable/24119959 (accessed 06.04.2019). (In Eng.)

[12] Liu H., Morstatter F., Tang J., Zafarani R. The good, the bad, and the ugly: uncovering novel research opportunities in social media mining. *International Journal of Data Science and Analytics*. 2016; 1(3-4):137-143. (In Eng.) DOI: 10.1007/s41060-016-0023-0

[13] Thomsen J.G., Ernst E., Brabrand C., Schwartzbach M. WebSelF: A Web Scraping Framework. In: Brambilla M., Tokuda T., Tolksdorf R. (eds). *Proceedings of the 12th international conference on Web Engineering (ICWE'12)*. Springer-Verlag, Berlin, Heidelberg. 2012; 347-361. (In Eng.) DOI: 10.1007/978-3-642-31753-8_28

[14] Salerno J.J., Boulware D.M. Method and apparatus for improved web scraping. United States of America, 2003. Patentnr. US 7072890 B2 Available at: https://patents.google.com/patent/US7072890B2/en (accessed 06.04.2019). (In Eng.)

[15] Thomas D.M., Mathur S. Data Analysis by Web Scraping using Python. *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. Coimbatore, India. 2019; 450-454. (In Eng.) DOI: 10.1109/ICECA.2019.8822022

[16] Trifa A., Sbaï A.H., Chaari W.L. Evaluate a Personalized Multi Agent System through Social Networks: Web Scraping. *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. Poznan. 2017; 18-20. (In Eng.) DOI: 10.1109/WETICE.2017.14

[17] Marques P. et al. Detecting Malicious Web Scraping Activity: A Study with Diverse Detector. *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*. Taipei, Taiwan. 2018; 269-278. (In Eng.) DOI: 10.1109/PRDC.2018.00049

[18] Raulamo-Jurvanen P., Kakkonen K., Mäntylä M. Using Surveys and Web-Scraping to Select Tools for Software Testing Consultancy. In: Abrahamsson P., Jedlitschka A., Nguyen Duc A., Felderer M., Amasaki S., Mikkonen T. (eds). *Product-Focused Software Process Improvement. PROFES 2016. Lecture Notes in Computer Science*. Springer, Cham. 2016; 10027:285-300. (In Eng.) DOI: 10.1007/978-3-319-49094-6_18

[19] Budiarti R. P. N., Widyatmoko N., Hariadi M., Purnomo M.H. Web scraping for automated water quality monitoring system: A case study of PDAM Surabaya. *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. Lombok. 2016; 641-648. (In Eng.) DOI: 10.1109/ISITIA.2016.7828735

[20] Boeing G., Waddell P. New Insights into Rental Housing Markets across the United States: Web Scraping and Analyzing Craigslist Rental Listings. *Journal of Planning Education and Research.* 2016; 37(4):457-476. (In Eng.) DOI: 10.1177/0739456X16664789

[21] Adamuz P.L. Development of a generic test-bed for web scraping. Barcelona: European Education and Training Accreditation Center, 2015.

Submitted 06.04.2019; revised 20.05.2019;
published online 25.07.2019.
Поступила 06.04.2019; принята к публикации 20.05.2019;
опубликована онлайн 25.07.2019.

Об авторах:

Москаленко Алексей Анатольевич, студент, факультет вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова (119991, Россия, г. Москва, ГСП-1, Ленинские горы, д. 1), ORCID: http://orcid.org/0000-0002-3719-9856, alekmosk25@gmail.com

Лапонина Ольга Робертовна, научный сотрудник лаборатории открытых информационных технологий, факультет вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова (119991, Россия, г. Москва, ГСП-1, Ленинские горы, д. 1), ORCID: http://orcid.org/0000-0001-5903-6858, laponina@oit.cmc.msu.ru

Сухомлин Владимир Александрович, заведующий лабораторией открытых информационных технологий, факультет вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова (119991, Россия, г. Москва, ГСП-1, Ленинские горы, д. 1), доктор технических наук, профессор, Президент Фонда «Лига интернет-медиа», ORCID: http://orcid.org/0000-0001-9468-7138, sukhomlin@mail.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:
Alexey A. Moskalenko, undergraduate student, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia), ORCID: http://orcid.org/0000-0002-3719-9856, alekmosk25@gmail.com
Olga R. Laponina, researcher, Laboratory of Open Information Technologies, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia), ORCID: http://orcid.org/0000-0001-5903-6858, laponina@oit.cmc.msu.ru
Vladimir A. Sukhomlin, Head of the Laboratory of Open Information Technologies, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia), Dr.Sci. (Technology), Professor, President of the Fund "League Internet-Media", ORCID: http://orcid.org/0000-0001-9468-7138, sukhomlin@mail.ru

All authors have read and approved the final manuscript.