BRIEFINGS IN BIOINFORMATICS. VOL 15. NO 5. 788–797
Advance Access published on 30 April 2013
doi:10.1093/bib/bbt026

Web scraping technologies in an API world

Daniel Glez-Peña, Anália Lourenço, Hugo López-Fernández, Miguel Reboiro-Jato and Florentino Fdez-Riverola

Submitted: 6th February 2013; Received (in revised form): 28th March 2013

Abstract

Web services are the *de facto* standard in biomedical data integration. However, there are data integration scenarios that cannot be fully covered by Web services. A number of Web databases and tools do not support Web services, and existing Web services do not cover for all possible user data demands. As a consequence, Web data scraping, one of the oldest techniques for extracting Web contents, is still in position to offer a valid and valuable service to a wide range of bioinformatics applications, ranging from simple extraction robots to online meta-servers. This article reviews existing scraping frameworks and tools, identifying their strengths and limitations in terms of extraction capabilities. The main focus is set on showing how straightforward it is today to set up a data scraping pipeline, with minimal programming effort, and answer a number of practical needs. For exemplification purposes, we introduce a biomedical data extraction scenario where the desired data sources, well-known in clinical microbiology and similar domains, do not offer programmatic interfaces yet. Moreover, we describe the operation of WhichGenes and PathJam, two bioinformatics meta-servers that use scraping as means to cope with gene set enrichment analysis.

Keywords: *Web scraping; data integration; interoperability; database interfaces*

BACKGROUND

Currently, biomedical research is highly dependent on Web resources and tools, such as online data repositories, online and downloadable data analysis tools, scientific literature catalogues, text mining systems and visualization artefacts. The PubMed literature search service, the Ensembl genome browser [1], the KEGG [2] and BioCyc pathway databases [3], together with the National Center for Biotechnology Information (http://www.ncbi.nlm.nih.gov/) and the European Bioinformatics Institute (http://www.ebi.ac.uk/) portals, are examples of some of the Web resources that many biologists use on a daily basis. However, the inventory of biomedical resources is continually growing, changing and evolving. The 19th annual Database Issue of *Nucleic Acids Research* (NAR) journal lists 1380 databases, featuring 92 new entries in 2011 and 100 articles reporting recent updates to existing databases in NAR and related journals [4].

Bioinformatics tools are expected to manage and take advantage of this plethora of resources in the best possible way, i.e. looking for and extracting the contents of interest for a given application,

Corresponding author. Florentino Fdez-Riverola, Escuela Superior de Ingeniería Informática, Edificio Politécnico. Campus Universitario As Lagoas s/n, 32004 Ourense, Spain. Tel.: +34 988387015; Fax: +34 988387001; E-mail: riverola@uvigo.es

Daniel Glez-Peña is an Associate Professor at the Department of Computer Science, University of Vigo. His research interests include knowledge integration techniques applied to DNA microarray analysis, as well as bioinformatics Web services and meta-servers development.

Anália Lourenço is a Faculty Professor at the Department of Computer Science, University of Vigo, and a Research Associate of the Centre of Biological Engineering, University of Minho. Her research interests include genome-scale model reconstruction, biological network analysis and development of large-scale bio-data analysis applications.

Hugo López Fernández is a PhD fellow in the Computer Science Department at the University of Vigo. He is currently working on bioinformatics annotation tools using biomedical ontologies, as well as mass spectrometry data analysis.

Miguel Reboiro-Jato is an Associate Professor at the Department of Computer Science, University of Vigo. His research interests include knowledge-based ensemble learning applied to DNA and mass spectrometry data analysis, as well as Web services and meta-servers development.

Florentino Fernández-Riverola is the leader of the Next Generation Computer Systems Group at the University of Vigo, involved in leading projects and PhD thesis in the field of bioinformatics since 2004.

within a reasonable timeframe and without consuming too many resources. In the past, bioinformaticians wrote software to automatically extract from Web sites information that was originally designed for human consumption, the so-called Web data scraping or, more generally, screen-scraping. This kind of programming was fairly simple as scrapers work over the underlying object structure of the HTML-based application, namely, the Document Object Model of the HTML. However, it implied site-specific programming and did not comply with (expectable) changes in the HTML source [5]. These limitations, and the ever-growing number of resources made available, led to the creation of agreed-on data APIs, the so-called programmatic interfaces, which provide for a much more robust structure to download and interconnect large sets of heterogeneous information.

Now, Web services are the *de facto* standard for biomedical data interoperability, and Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) are the two major implementation approaches [6]. The most important biomedical servers offer this operation mode, and the NAR journal has acknowledged their relevance, dedicating an annual issue to disseminate the Web services available [7]. Although the proliferation of Web services has reduced the need of Web data scrapers, there are still scenarios where they are still useful, namely, whenever programmatic interfaces are not available, e.g. there still remain domains with little interoperability standards or relevant Web services [8]; programmatic interfaces are insufficient, i.e. the existing APIs do not give access to the desired tool or data [9] and programming costs related to learning a set of (possibly) heterogeneous APIs [10] are not justified, e.g. when the retrieval is to be performed only once for matters of prototyping or source evaluation. Moreover, bioinformatics data integration frameworks often include a scraping-alike option in anticipation to user requests for uncommon and sporadic data sources.

Just for illustrative purposes, bioinformatics frameworks such as Firegoose [11] and tools such as Protein Information Crawler [12], DrugBank [13], ChemSpider [14], BioSpider [15], OReFil [16] and MEDPIE [17] acknowledge (or had acknowledged at some point) the use of Web data scraping. Moreover, different examples of Web data scrapping can be found in recent domain-specific applications across Life Sciences, such as in Biotechnology and Bioengineering [18–21], Genetics [22–24], Molecular Biology [25,26], Crystallography [27] and Medicine [28–31].

Given the heterogeneous nature of the applications potentially in need of Web data scraping, the aim of this article is twofold: to identify the main artefacts to be considered in the implementation of a Web scraper and to pinpoint how existing scraping tools and frameworks can be of use to current biomedical applications. As examples, we introduce a data extraction scenario on antimicrobial susceptibility and novel drugs and report the case of WhichGenes and PathJam, two meta-servers operating in the field of functional genomics.

The organization of the article goes as follows. The second section defines the Web scraping technique, introduces its most common tasks and overviews available solutions to speed up this kind of programming. The third section describes the scraping pipelines of the proposed biomedical case studies. Finally, the fourth section discusses the utility of Web scraping today.

BUILDING WEB DATA SCRAPERS

Generally, Web data scraping can be defined as the process of extracting and combining contents of interest from the Web in a systematic way. In such a process, a software agent, also known as Web robot, mimics the browsing interaction between the Web servers and the human in a conventional Web traversal. Step by step, the robot accesses as many Web sites as needed, parses their contents to find and extract data of interest and structures those contents as desired.

Web scraping APIs and frameworks address the most common tasks Web data scrapers get involved in to accomplish particular retrieval goals, as described in the following text:

- Site access: The Web data scraper establishes communication with the target Web site through the HTTP protocol, a stateless text-based Internet protocol that coordinates the request–response transactions between a client, typically a Web browser, and a Web server. In HTTP, the most frequent request 'methods' are GET, used in resource requests, and POST, used in form submission and file uploading. The 'User-Agent' is also an important request header, because the server looks into it to find out what kind of program is accessing its contents (browser versus

robot), and eventually differentiate user responses. Furthermore, like any other Web robot, Web data scrapers are expected to conform to the 'terms of use' of the site as described in its 'robots.txt' file (a file hosted at the server, which states which resources should not be accessed by automatic procedures), and should schedule retrieval tasks carefully to avoid server overloading.

- HTML parsing and contents extraction: Once the HTML document is retrieved, the Web data scraper may extract the contents of interest. For this purpose, regular expression matching, alone or in combination with additional logic, is widely adopted. As an alternative, there are HTML parsing libraries (working over the Document Object Model structure of the Web pages), and selector-based languages, such as XPath (http://www.w3.org/TR/xpath20/) and the CSS selector syntax. As a general guideline, it is recommended to keep matching expressions as general as possible, to make robots less vulnerable to changes in the HTML document. Scraping is more robust when the site implements semantic markup, such as Microformats (http://microformats.org) or Microdata (http://www.whatwg.org).
- Output building: The main goal is to transform the extracted contents into a structured representation that is suitable for further analysis and storage. Although this final step is marginal to Web scraping, some tools are aware of result post-processing, providing for in-memory data structures and text-based solutions, such as strings or files (typically XML or CSV files).

Software for Web data scraping

Existing approaches to implement a Web data scraper can be structured into three main categories: (i) libraries for general-purpose programming languages, (ii) frameworks and (iii) desktop-based environments.

Libraries

One of the most common approaches used by bioinformaticians consists in constructing their own Web data scrapers using the programming language they are most familiar with. In this case, the logic behind the robot and the final result are implemented as conventional software programs, i.e. using the control and data structures of the language. Usually, third-party libraries grant access to the site by implementing the client side of the HTTP protocol, whereas the retrieved contents are parsed using built-in string functions, such as regular expression matching, tokenization and trimming. Third-party packages may also provide for more sophisticated parsing, such as HTML tree building and XPath matching.

One of the most popular site access libraries is libcurl (http://curl.haxx.se/). It supports the major features of the HTTP protocol, including SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form-based upload, proxies, cookies and HTTP authentication. Moreover, it has useful bindings to many programming languages.

Perl, which is one of the programming languages most widely used in bioinformatics, incorporates the *WWW::Mechanize* Web automation module (http://search.cpan.org/~jesse/WWW-Mechanize-1.72/lib/WWW/Mechanize.pm). This module is able to interact with Web links and forms without additional parsing, and provides support for HTTPS, cookie management, HTTP authentication and history management, among others. Moreover, it allows the use of XPath through additional modules.

In Java, the Apache HttpClient package (http://hc.apache.org/httpcomponents-client-ga/index.html) emulates HTTP main features, i.e. all request methods, cookies, SSL and HTTP authentication, and can be combined with HTML parsing libraries such as jsoup (http://jsoup.org/). Java also supports XPath and provides several HTML cleaning libraries, such as htmlcleaner (http://htmlcleaner.sourceforge.net). Similarly, the BeautifulSoup (http://www.crummy.com/software/BeautifulSoup/) is a Python HTML parsing library, which can be combined with language native support for HTTP connections. Moreover, in Unix-like environments, by simply piping operating system command-line programs inside shell scripts, programmers are able to create Web data scrapers in one or few lines of code. Programs like curl (libcurl) and wget (http://www.gnu.org/software/wget/) implement the HTTP client layer, while utilities such as grep, awk (http://www.gnu.org/software/gawk/), sed (http://www.gnu.org/software/sed/) and cut and paste can be used to parse and transform contents conveniently.

In the case of server side robots, typically running inside Web applications, a 100% compatible technology with the programming language (typically PHP, Perl or Java) is recommended.

Frameworks

Using a general-purpose language to create robots has some drawbacks. Often, several libraries need to be

Table 1: Open-source Web scraping libraries and frameworks

	Type C: HTTP client P: Parsing F: Framework	Domain-specific language	API/stand alone	Language	Extraction facilities R: Regular expressions H: HTML parsed tree X: XPath C: CSS selectors
UNIX shell (curl/wget, grep, sed, cut, paste, awk)	CP	No	SA	bash	R
Curl/libcurl	C	No	Both	C + bindings	
Web-Harvest	F	Yes	Both	Java	RX
Jsoup	CP	No	API	Java	HC
HttpClient	C	No	API	Java	
jARVEST	F	Yes	Both	JRuby/Java	RXC
WWW::Mechanize	CP	No	API	Perl	RX
Scrapy	F	No	Both	Python	RX
BeautifulSoup	P	No	No	Python	H

We have selected several available Web scraping packages oriented to programmers. There are six libraries implementing an HTTP client (C) and/or HTML parsing/extraction (P) and three frameworks (F). Web-Harvest and jARVEST frameworks present a domain-specific language for defining robots, based on XML and Ruby, respectively. For all the analyzed alternatives, we report their extraction facilities, including regular expressions (R), HTML parsed tree (H), XPath expressions (X) and CSS Selectors (C).

integrated, in particular one for Web access and others to parse and extract content from the HTML documents. Furthermore, robots are known to be weak pieces of software, which are considerably affected by changes in the HTML of the accessed resources, and thus require continuous maintenance. In compiled languages, such as Java, any change in the implementation of the robot forces re-recompilation and even the re-deploy of the entire application.

Scraping frameworks present a more integrative solution. For example, Scrapy (http://scrapy.org) is a powerful Web scraping framework for Python, where robots are defined as classes inheriting from BaseSpider class, which defines a set of 'starting urls' and a 'parse' function called at each Web iteration. Web pages are automatically parsed and Web contents are extracted using XPath expressions.

Other frameworks present domain-specific languages (DSL), which are specific programming languages designed for a particular domain and, therefore, robots are treated as independent and external artefacts. An example of this is Web-Harvest (http://web-harvest.sourceforge.net/), a Web data scraping framework for Java. Here, Web extraction processes are described in XML (with the help of a visual environment) and are composed of several 'pipelines', which can include procedural instructions, such as variable definitions and loops, as well as many primitives, such as 'http' (to retrieve Web contents), 'html-to-xml' (to clean HTML) and 'xpath' to extract content. Another example of a Java Web data scraping framework is jARVEST (http://sing.ei.uvigo.es/jarvest), which also defines a DSL, but uses JRuby for a more compact robot implementation.

Table 1 summarizes and compares several of the most popular open-source Web scraping libraries and frameworks.

Desktop-based environments

Desktop applications attend to the needs of layman programmers. This kind of tools is empowered by graphical design environments, which facilitate the creation and maintenance of robots. Usually, the software includes an integrated browser, where the user can navigate to the target Web and interactively select the elements of the page to be extracted, avoiding any specification of regular expressions, XPath queries or other technicalities. In addition, modules are available to build multiple kinds of output, such as files in CSV, Excel and XML format, and insertions into databases.

The major drawbacks of desktop solutions are the commercial distribution and limited API access, which make it difficult to embed these scrapers inside other programs (which it is often a requirement). In Table 2, seven common desktop-based Web scraping tools are compared.

ILLUSTRATIVE SCENARIOS OF WEB SCRAPING

To conveniently illustrate the importance of the previously exposed concepts, this section introduces two common, but different, biomedical scenarios in

Table 2: Desktop-based Web scraping solutions

	IrobotSoft[a]	Visual Web Ripper[b]	Newbie[c]	Mozenda[d]	Screen-scraper[e]	WebSundew[f]	FMiner[g]
Software type							
License	Freeware	Commercial	Commercial	Commercial	Freeware (Basic edition)	Commercial	Commercial
Open source	No	No	No	No	No	No	No
Platforms	Win	Win	Win	Win	Win Linux Mac	Win	Win
Site access							
Form POST	Yes	Yes	Yes	Yes	Yes	No	Yes
Session	Yes	Yes	Yes	Yes	Yes	No	Yes
Conf. user agent	IE	IE and 2 internal UAs	IE Firefox	IE	No	Firefox and I internal UA	No
Iteration over pages	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Anonymizer Proxies	Yes	Yes	N/A	No	No	N/A	Yes
Formats							
Input formats	.irb	.rip	Webpages URL list	.xml	.sss	.zws	.sep
Output formats	Text CSV XML DB	CSV XML DB Excel	Text Excel DB	CSV TSV XML Excel	Text CSV DB	CSV XML Excel	CSV Excel DB
Robot file format	.irb	.rip	.nbs scripts	.xml	.sss	.zws	.sep
Runtime							
Multi-threading	Yes	Yes	Yes	No	Yes	Yes	Yes
Progressive results	Yes	No	Yes	Yes	Yes	Yes	Yes
Design environment							
GUI-based designer	Yes	Yes	Yes	Yes	Yes	Yes	Yes
API access	No	Yes	Yes	Yes	Yes	Yes	No
Scriptable	Yes	Limited	Yes	No	Yes	N/A	No

Comparison of functionalities of different desktop-based scraping solutions. We have selected several features to evaluate the tools, including software license, supported platforms, site access capabilities, runtime aspects and robot design possibilities. [a]http://www.irobotsoft.com. [b]http://www.visualwebripper.com. [c]http://www.newbielabs.com. [d]http://www.mozenda.com. [e]http://www.screen-scraper.com. [f]http://www.websundew.com. [g]http://www.fminer.com.

need of Web data scraping support. In the first scenario, data are compiled from public Web sites that do not yet provide APIs to download and interconnect data. The second scenario shows the resourcefulness of Web data scraping in third-party data integration, also known as meta-server deployment.

Screening information on antimicrobial peptides

The increasing resistance of microbial pathogens to conventional therapeutics is promoting extensive antimicrobial screening and susceptibility testing. Among other applications, automated retrieval tools can be useful for designing susceptibility tests (e.g. choosing the most interesting products and conditions to be tested), implementing vertical search engines on antimicrobial products, reconstructing cellular models (e.g. gene–drug and drug–drug interaction networks) and developing biomedical text mining tools in support of related database curation workflows. Here, we present an example of the practical usefulness of screen scraping, given that most of the data sources relevant to the field do not implement Web services yet.

Starting with a target pathogenic species, *Pseudomonas aeruginosa*, we search for natural compounds and effective concentrations with antimicrobial activity against that species. Three main databases in the field are involved: the Antimicrobial Peptide Database [32,33] and the Collection of Anti-Microbial Peptides [34] provide details on the bioactivity of natural peptides, and the database of the European Committee on Antimicrobial Susceptibility Testing (http://www.eucast.org/mic_distributions/) provides data on the minimum inhibitory concentration (MIC), i.e. the lowest concentration of the antimicrobial that will inhibit the visible

growth of the microorganism. The aim of this scraping process is to provide an integrated view of the information compiled about antimicrobial products known to target *P. aeruginosa*.

Web data scrapers were programmed to retrieve European Committee on Antimicrobial Susceptibility Testing's MIC values and anti-microbial peptide (AMP) information for *P. aeruginosa* (Figure 1). In Collection of Anti-Microbial Peptides, GI and UniProt identifiers are used to link to sequence and function information on the peptides, whereas scientific literature is linked through PubMed identifiers. Using protein identifiers, the scraper is able to get from Antimicrobial Peptide Database, additional information on the mode of action of the peptide as well as indicators of bioactivity, such as structure, charge and Boman index.

The output file, as to be presented to the user, summarizes existing information about the antimicrobial activity of antibiotics and natural peptides against *P. aeruginosa*, as described in the following text:

- Peptide names, which are important to conciliate and standardize terminology, and thus promote multi-source data integration.
- AMP source, usually the taxonomic name of the plants or animals from where the AMP is extracted.
- AMP activity, designated by terms such as antibacterial (even specifics on gram-positive and gram-negative bacteria), antifungal, cancer cells, etc.
- MIC values for the antimicrobial products found to target *P. aeruginosa*.
- Database identifiers, which enable source cross-linking as well as access to protein sequence and similar attributes (UniProt SWISS-PROT [35] and NCBI Protein [36]).
- Bibliographic references.

This Web data scraping workflow was implemented using jARVEST and the corresponding code can be found in Supplementary Material 1.

Building bioinformatics meta-servers

The WhichGenes meta-server [37] retrieves lists of genes that are related to diseases, involved in metabolic pathways, annotated with GO terms, target of microRNAs and so on. WhichGenes meta-server enables the user to retrieve gene lists from different data sources in a uniform way, and may further operate over the gene lists, by performing unions, intersections and differences, to generate new hypotheses.

WhichGenes works in a federated mode, obtaining gene lists by querying third-party resources. For each resource, a specific adapter takes a query as input and returns a gene list. There are two types of queries: free text queries (e.g. a disease name) and constrained queries (e.g. GO terms or pathways). In the latest, the retrieval of possible query terms is also responsibility of the corresponding resource adapter. When necessary, the gene names are converted from their source namespace to HGNC symbol or MGI symbol for human and mouse genes, respectively.

From the set of resources involved in the operation of WhichGenes, data extraction on 7 of 11 sources of information is still based on Web scraping (Table 3). As the project runs scraping robots on every user query, a monitoring subsystem is in charge of executing tests periodically to detect unresponsive services and obsolete robots (i.e. those giving unexpected or empty results).

PathJam is another operational meta-server [38] devoted to pathway database integration. It enables the functional analysis of gene lists, typically coming from microarrays differential expression analyses, against pathway data from three databases: KEGG, Reactome [39] and NCI-PID [40]. Following a data warehousing architecture, PathJam builds a pathway-gene mapping periodically. Additionally, the meta-server maintains a gene dictionary (from Ensembl) to support the correct integration of pathway databases and user gene lists, independently of the namespaces used. Currently, only the NCI-PID database is still accessed by Web data scraping, as the rest of the databases already offer programmatic APIs.

DISCUSSION

Nowadays, there is a diversity of content-bearing Web resources that complements a variety of biomedical needs. The challenge lays on dealing with these volumes of information, enabling valuable and valid information screening and data integration. Often, researchers need to consult several independent and heterogeneous resources to solve specific biological questions. Yet, manual screening is time-consuming, demands expertise in the field and, still, it is prone to miss valuable details. Information is typically scattered across institutional and laboratory

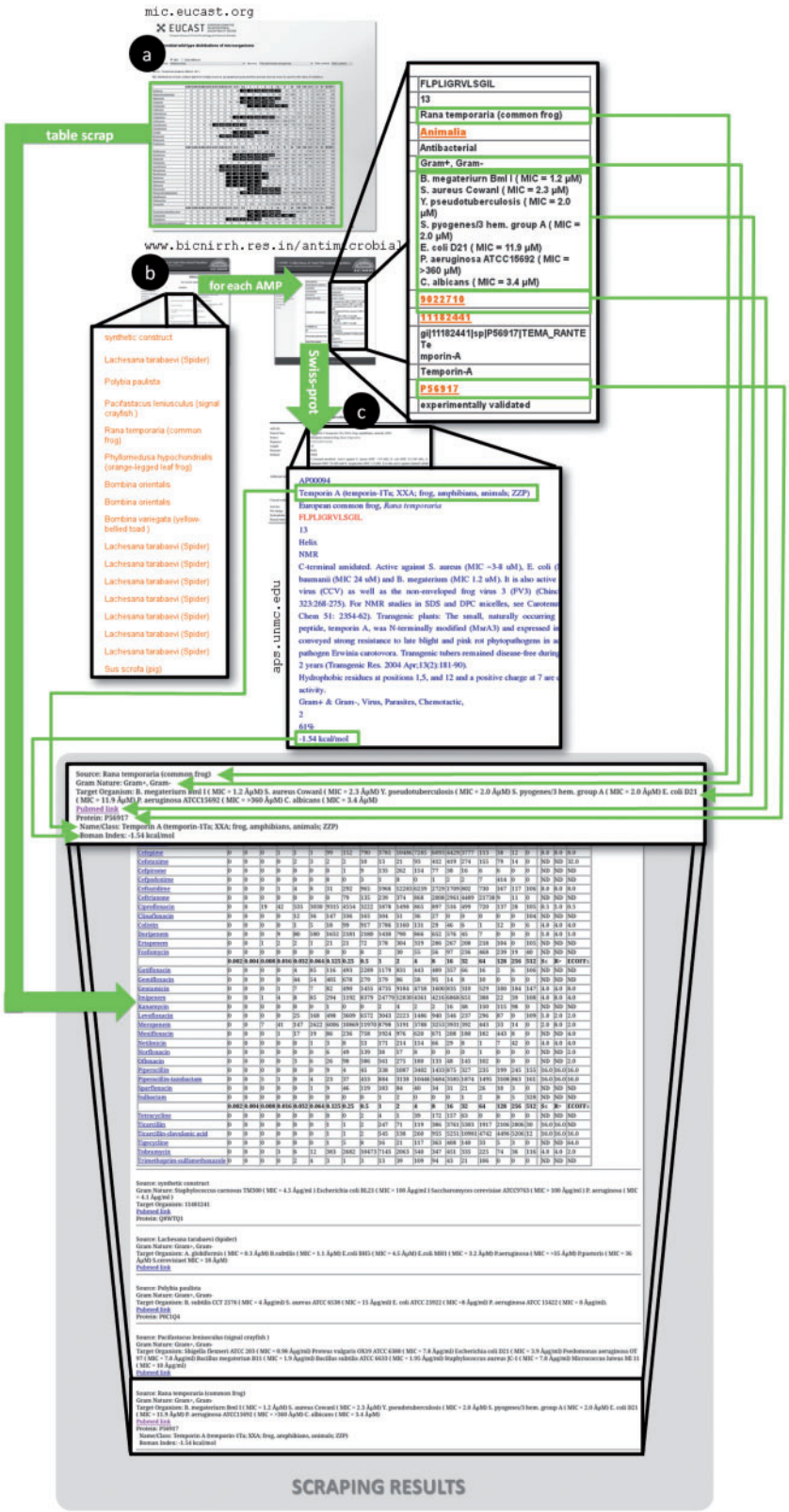

Figure 1: Overview of a data scraping workflow to retrieve European Committee on Antimicrobial Susceptibility Testing's (EUCAST) MIC values and Collection of Anti-Microbial Peptides' (CAMP) and Antimicrobial Peptide Database's (APD) AMP information. (a) The robot scrapes an HTML table containing the EUCAST's MIC values directly from a known URL. (b) It retrieves AMP information starting from the CAMP database, where a list of AMPs for *aeruginosa* can be found, getting (c) detailed information and cross-linking with SWISSPROT IDs to the CAMP database to obtain additional data.

Table 3: Third-party resources accessed by WhichGenes using Web data scraping techniques

WhichGenes provider	Site accessed	Robot main tasks
GeneCards disease genes [41]	www.genecards.org	Perform a query by using the 'advanced search form', passing the disease entered by the user at WhichGenes. Parse the resulting HTML.
Gene Ontology annotated genes [42]	www.berkeleybop.org/goose (GO mirror)	Access the 'GO Online SQL environment' to search genes annotated with the GO terms selected by the user at WhichGenes. Simple parsing of the resulting plain-text tabular file.
MSigDB Positional gene sets [43]	www.broad.mit.edu/gsea/msigdb	Access the MSigDB Web page to (i) extract the current available positional gene sets and (ii) retrieve the gene set selected by the user at WhichGenes. Both tasks include HTML parsing. In addition, MSigDB requires user identification, before accessing the files (giving a valid e-mail). The robot posts the WhichGenes authors' e-mail on their behalf.
TargetScan microRNA targets [44]	www.targetscan.org	Perform a query by using the TargetScan search form at the home page, passing the microRNA ID given by the user at WhichGenes. Parse the resulting HTML table.
miRBase microRNA targets [45]	microrna.sanger.ac.uk/cgi-bin/targets/v5/download.formatter.pl	Access the 'download.formatter.pl' URL at miRBase, which dumps the genes that are target of a user-given microRNA at WhichGenes. Simple parsing of the resulting plain-text tabular file.
CancerGenes gene lists [46]	cbio.mskcc.org/CancerGenes	Access the CancerGenes Web page to (i) extract the current available source gene lists and (ii) retrieve the gene list selected by the user at WhichGenes. Both tasks include HTML parsing.
IntAct interaction genes [47]	www.ebi.ac.uk/intact/export	Access the 'export' URL at IntAct, which dumps the related proteins that interact with a user-given gene symbol. Simple parsing of the resulting plain-text tabular file.

WhichGenes accesses seven third-party resources by using Web data scraping, mainly invoking URLs through GET methods. Usually, URLs correspond to search engines at the provider that gives (i) HTMLs as output that are subsequently parsed by the robot, or (ii) tabular data files that are easy to transform.

Web sites, and may be highlighted only in journal articles or conference proceedings. Moreover, the heterogeneity of data formats and data types involved is significant, and data integration and interpretation are highly dependent on the biological question. Therefore, databases and tools need to be equipped to convey with the design and execution of automated data workflows, supporting hypothesis testing and knowledge acquisition within various scopes of application.

Biomedical data resources are in general committed with the open science data movements, which promote the public dissemination of scientific results and data. The value of the data therefore increases with greater openness. Many sites are progressively incorporating emerging and easy-to-use semantic markup standards, such as Microformats, Microdata or, more recently, Facebook's Open Graph protocol (http://developers.facebook.com/docs/opengraph). Meanwhile, Web services are the standard and recommended way to enable external access to biomedical databases and services. Notwithstanding, Web services are not sufficient to grant full biomedical data interoperability and integration. Owing to various development constraints (such as technical expertise, costs, evaluation of to-be-expected functionalities and establishment of the desired quality of service), it is not a common practice to make available public APIs for Web databases and servers in their early years. Typically, Web site creators are focused on both providing high-quality contents through expert manual curation, and deploying online search functions targeting the interests of biomedical practitioners. In fact, standard Web services are usually developed only for mature databases and servers, with a considerable volume of traffic and well-profiled usage expectations.

Furthermore, the costs associated with learning to operate a programmatic interface (most likely several, to meet an integrative problem perspective) should not be dismissed and should be assessed in terms of the desired time of response (i.e. how soon data are needed), the nature of application (e.g. the bioinformatician may not be interested in publishing scrapped data but rather to use it in internal processes and, therefore, may wish to keep deployment as simple as possible) and the longevity of the data

extraction task (i.e. one time task versus recurring task). As a consequence, it is only fair to acknowledge that Web data scraping may still help in many daily, one-time or private consumption biomedical information extraction tasks as well as intervene in broader projects, such as in the construction of meta-servers and other integrative biomedical resources.

The most popular way to build Web robots is to use third-party libraries, often a tandem of a site access library and an HTML parsing library, which represents a small learning curve. Despite implying a more pronounced learning curve, scraping frameworks provide for a comprehensive coverage of the scraping lifecycle and, in some cases, DSLs to facilitate the maintenance of robots. In addition, there are also commercial graphical desktop environments, suitable for less experienced users and to swift simpler deployments.

Independently of the implementation, Web scraping developers should take into consideration the legal and policy issues and program Web scrapers compliantly. Although the legal implications are not totally clear in all cases and countries, developers should take into consideration the 'terms of use', namely, preventing copyright infringement, balancing the number and frequency of the requests and skipping resources that are tagged as not to be scraped.

SUPPLEMENTARY DATA

Supplementary data are available online at http://bib.oxfordjournals.org/.

Key Points

- Web services or API access are not always available in online biomedical resources.
- Web scraping is still used in many bioinformatics projects, ranging from private, one-time use data generation to regular feeding of online meta-servers.
- There are multiple software solutions supporting Web scraping development, including libraries, frameworks and desktop-based applications.

FUNDING

This work was partially funded by (i) the [TIN2009–14057-C03–02] project from the Spanish Ministry of Science and Innovation, the Plan E from the Spanish Government and the European Union from the European Regional Development Fund (ERDF), (ii) the Portugal–Spain cooperation action sponsored by the Foundation of Portuguese Universities [E 48/11] and the Spanish Ministry of Science and Innovation [AIB2010PT-00353] and (iii) the Agrupamento INBIOMED [2012/273] from the DXPCTSUG (Dirección Xeral de Promoción Científica e Tecnolóxica do Sistema Universitario de Galicia) from the Galician Government and the European Union from the ERDF unha maneira de facer Europa. H.L.F. was supported by a pre-doctoral fellowship from the University of Vigo.

References

1. Flicek P, Amode MR, Barrell D, *et al*. Ensembl 2012. *Nucleic Acids Res* 2012;**40**:D84–90.
2. Kanehisa M, Goto S, Sato Y, *et al*. KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Res* 2012;**40**:D109–14.
3. Caspi R, Altman T, Dale JM, *et al*. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Res* 2010;**38**:D473–9.
4. Galperin MY, Fernández-Suárez XM. The 2012 Nucleic acids research database issue and the online molecular biology database collection. *Nucleic Acids Res* 2012;**40**:D1–8.
5. Stein L. Creating a bioinformatics nation. *Nature* 2002;**417**:119–120.
6. Stockinger H, Attwood T, Chohan SN, *et al*. Experience using web services for biological sequence analysis. *Brief Bioinform* 2008;**9**:493–505.
7. Benson G. Editorial. *Nucleic Acids Res* 2012;**40**:W1–2.
8. Katayama T, Arakawa K, Nakao M, *et al*. The DBCLS BioHackathon: standardization and interoperability for bioinformatics web services and workflows. *J Biomed Semantics* 2010;**1**:8.
9. Day R, Lisovich A. DAVIDQuery: Retrieval from the DAVID bioinformatics data resource into R. R package version 1.18.0. 2010. http://bioconductor.org/packages/release/bioc/vignettes/DAVIDQuery/inst/doc/DAVIDQuery.pdf (2 February 2013, date last accessed).
10. Goble C, Stevens R. State of the nation in data integration for bioinformatics. *J Biomed Inform* 2008;**41**:687–93.
11. Bare JC, Shannon PT, Schmid AK, *et al*. The Firegoose: two-way integration of diverse data from different bioinformatics web resources with desktop applications. *BMC Bioinformatics* 2007;**8**:456.
12. Mayer U. Protein Information Crawler (PIC): extensive spidering of multiple protein information resources for large protein sets. *Proteomics* 2008;**8**:42–4.
13. Wishart DS, Knox C, Guo AC, *et al*. DrugBank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Res* 2006;**34**:D668–72.
14. Williams A. *Using Text-Mining and Crowdsourced Curation to Build a Structure Centric Community for Chemists,* 2008. http://www.slideshare.net/AntonyWilliams/using-textmining-and-crowdsourced-curation-to-build-a-structure-centric-community-for-chemists-presentation/ (22 January 2013, date last accessed).

15. Knox C, Shrivastava S, Stothard P, *et al*. BioSpider: a web server for automating metabolome annotations. *Pac Symp Biocomput* 2007;145–56.
16. Yamamoto Y, Takagi T. OReFiL: an online resource finder for life sciences. *BMC Bioinformatics* 2007;**8**:287.
17. Benton A, Holmes JH, Hill S, *et al*. medpie: an information extraction package for medical message board posts. *Bioinformatics* 2012;**28**:743–4.
18. Yang Y, Wilson LT, Wang J. Development of an automated climatic data scraping, filtering and display system. *Comput Electron Agric* 2010;**71**:77–87.
19. Beran B, Piasecki M. Engineering new paths to water data. *Comput Geosci* 2009;**35**:753–60.
20. Johnson S. Design & Implementation of a Pipeline for High-throughput Enzyme Function Prediction [PhD dissertation]. Fairfax, Virginia: George Mason University for the Degree of Master of Science Bioinformatics, 2006.
21. Beran B, Goodall J, Valentine D, *et al*. Standardizing access to hydrologic data repositories through Web services. *International Conference on Advanced Geographic Information Systems Web Services, 2009 (GEOWS'09)*. Piscataway, NJ, USA: IEEE, 2009;64–67.
22. Wall DP, Pivovarov R, Tong M, *et al*. Genotator: a disease-agnostic tool for genetic annotation of disease. *BMC Med Genomics* 2010;**3**:50.
23. Springate D. Scraping organism metadata for Treebase repositories from GOLD using Python and R. Posted at David Springate's blog, 2012. http://daspringate.github.io/2013-04/scraping_metadata.html (16 April 2013, date last accessed).
24. National Heart, Lung and Blood Institute, National Human Genome Research Institute. SeattleSeq Annotation. http://snp.gs.washington.edu/SeattleSeqAnnotation129/HelpAbout.jsp (2 February 2013, date last accessed).
25. Ranzinger R, Herget S, Wetter T, *et al*. GlycomeDB—integration of open-access carbohydrate structure databases. *BMC Bioinformatics* 2008;**9**:384.
26. Verslyppe B, Kottmann R, De Smet W, *et al*. Microbiological Common Language (MCL): a standard for electronic information exchange in the Microbial Commons. *Res Microbiol* 2010;**161**:439–45.
27. Day NE. *Automated Analysis and Validation of Open Chemical Data*. [dissertation]. Cambridge, UK: University of Cambridge for the degree of Doctor of Philosophy. Unilever Centre from Molecular Science Informatics, Department of Chemistry, 2009.
28. Tenenbaum JD, Whetzel PL, Anderson K, *et al*. The Biomedical Resource Ontology (BRO) to enable resource discovery in clinical and translational research. *J Biomed Inform* 2011;**44**:137–45.
29. Inusah S, Sormani MP, Cofield SS, *et al*. Assessing changes in relapse rates in multiple sclerosis. *Mult Scler* 2010;**16**:1414–21.
30. Rajapakse M, Kanagasabai R, Ang WT, *et al*. Ontology-centric integration and navigation of the dengue literature. *J Biomed Inform* 2008;**41**:806–15.
31. Hill AW, Guralnick RP. Distributed systems and automated biodiversity informatics: genomic analysis and geographic visualization of disease evolution. *Inf Knowl* 2008;270–9.
32. Wang Z, Wang G. APD: the antimicrobial peptide database. *Nucleic Acids Res* 2004;**32**:D590–2.
33. Wang G, Li X, Wang Z. APD2: the updated antimicrobial peptide database and its application in peptide design. *Nucleic Acids Res* 2009;**37**:D933–7.
34. Thomas S, Karnik S, Barai RS, *et al*. CAMP: a useful resource for research on antimicrobial peptides. *Nucleic Acids Res* 2010;**38**:D774–80.
35. The UniProt Consortium Reorganizing the protein space at the Universal Protein Resource (UniProt). *Nucleic Acids Res* 2012;**40**:D71–5.
36. Wheeler DL, Barrett T, Benson DA, *et al*. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res* 2007;**35**:D5–12.
37. Glez-Peña D, Gómez-López G, Pisano DG, *et al*. WhichGenes: a web-based tool for gathering, building, storing and exporting gene sets with application in gene set enrichment analysis. *Nucleic Acids Res* 2009;**37**:W329–34.
38. Glez-Peña D, Reboiro-Jato M, Domínguez R, *et al*. PathJam: a new service for integrating biological pathway information. *J Integr Bioinform* 2010;**7**:147.
39. Croft D, O'Kelly G, Wu G, *et al*. Reactome: a database of reactions, pathways and biological processes. *Nucleic Acids Res* 2011;**39**:D691–7.
40. Schaefer CF, Anthony K, Krupa S, *et al*. PID: the pathway interaction database. *Nucleic Acids Res* 2009;**37**:D674–9.
41. Safran M, Chalifa-Caspi V, Shmueli O, *et al*. Human gene-centric databases at the Weizmann institute of science: GeneCards, UDB, CroW 21 and HORDE. *Nucleic Acids Res* 2003;**31**:142–6.
42. Gene Ontology Consortium. The Gene Ontology project in 2008. *Nucleic Acids Res* 2008;**36**:D440–4.
43. Subramanian A, Tamayo P, Mootha VK, *et al*. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl. Acad. Sci. U.S.A.* 2005;**102**:15545–50.
44. Lewis BP, Burge CB, Bartel DP, *et al*. Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microRNA targets. *Cell* 2005;**120**:15–20.
45. Griffiths-Jones S, Saini HK, van Dongen S, *et al*. miRBase: tools for microRNA genomics. *Nucleic Acids Res* 2008;**36**:D154–8.
46. Higgins ME, Claremont M, Major JE, *et al*. CancerGenes: a gene selection resource for cancer genome projects. *Nucleic Acids Res* 2007;**35**:D721–6.
47. Kerrien S, Alam-Faruque Y, Aranda B, *et al*. IntAct–open source resource for molecular interaction data. *Nucleic Acids Res* 2007;**35**:D561–5.