An End-User Pipeline for Scraping and Visualizing Semi-Structured Data over the Web

Gabriela Bosetti[1(✉)], Sergio Firmenich[1,2], Marco Winckler[3], Gustavo Rossi[1,2], Ulises Cornejo Fandos[1], and Előd Egyed-Zsigmond[4]

[1] LIFIA, Facultad de Informática, UNLP, 50th St. and 120th St., La Plata, Argentina
{gabriela.bosetti,sergio.firmenich,gustavo.rossi,
ulisescornejo.fandos}@lifia.info.unlp.edu.ar
[2] CONICET, Buenos Aires, Argentina
[3] i3S, Université Nice Sophia Antipolis, 2000, route des Lucioles, bât. Euclide B, BP 121, Sophia Antipolis, France
winckler@i3s.unice.fr
[4] Université de Lyon, LIRIS, INSA-Lyon, 7 Av. Jean Capelle, Villeurbanne, France
elod.egyed-zsigmond@insa-lyon.fr

Abstract. The Web is a vast source of semi-structured datasets that are made readily available to support the construction of new knowledge. Information visualization techniques have been demonstrated as a suitable alternative for allowing users to analyze and understand a large amount of data. However, the steps required for visualizing semi-structured data obtained from the Web is not straightforward, and it requires proper treatment before information visualization techniques could be applied. In this work, we present a visualization pipeline for describing the fundamental operations required for visualizing semi-structured data over the Web. We employ Web Scraping and Web Augmentation techniques for supporting interactive visualizations and solving tasks without changing the context of use of the data. Our approach is duly supported by a framework including scraping-, augmenting- and visualization-tools and it has been applied to different kinds of websites to demonstrate its validity and feasibility. Our ultimate goal is to expand the limits of our technology for improving the user interaction with websites and creating new experiences for a better understanding of large datasets.

Keywords: Infovis · Web augmentation · Web Scraping

1 Introduction

The Web is a massive source of public datasets. NetCraft[1] reported over 1.8 billion sites in the World at the beginning of 2017 and the NationalPost predicts[2] that by 2020, the amount of data produced annually will increase

[1] https://news.netcraft.com/archives/2017/01/12/january-2017-web-server-survey.html.

[2] https://nationalpost.com/news/big-data-and-analytics-taking-off-at-brocks-goodman-school-of-business.

M. Bakaev et al. (Eds.): ICWE 2019, LNCS 11496, pp. 223–237, 2019.
https://doi.org/10.1007/978-3-030-19274-7_17

4,300%. The Web not only made easier access to raw data but also made it available to everyone. In a recent survey published by data.world[3], 63% of citizens explore and interact with data to achieve a broad spectrum of tasks. The Broadband Commission for Sustainable Development (set up by the ITU and the UNESCO) estimates that the number of Internet users will represent half of the world's population at the end of 2019 at least [1]. In the context of the increasing amount of data, information visualization might play a role in helping many users to understand data. Visualization is indeed an important aspect of data analysis that allows conveying information in a visual format highlighting patterns, trends, and correlations among data [2].

Information visualization techniques are powerful tools specifically designed to support the exploration and analysis of large datasets, helping users to deal with complex decision-making tasks [3,4]. Information visualization techniques might improve both users' cognitive abilities and users' performance with tasks by relieving the working memory and improving decision accuracy, even on elderly people [5]. Many visualization techniques are intended to be used by specialized users (such as system administrators Mahendiran et al. [6]) but more and more often information visualization techniques (such as CivilAnalysis [7]) are developed to larger audiences.

Currently, more and more Web sites embed information visualization techniques to present their data in context. Nonetheless, this practice is not widespread and most Web sites still only display semi-structured data. The operations required for visualizing semi-structured data obtained from the Web is not straightforward, and it requires proper treatment before information visualization techniques could be applied. While the questions related to the design are central to the development of information visualization techniques, this paper is interested in the process, the so-called visualization pipeline, that allows transforming semi-structured data into graphical representations.

This paper investigates problems and possible solutions to build visualizations for helping end-users to analyze datasets available over the web. Our ultimate goal is to develop a technology that allows end-users to collect and visualize semi-structured data directly over the web site that publish the datasets. As we shall see, the answer to this problem is intimately associated with the many operations along the visualization pipeline. Hereafter, we present an approach and a tool that combines Web Scraping, Web Augmentation, and Information Visualization techniques. Moreover, we evaluate the validity and feasibility of the tools by running them over different kinds of websites.

2 Background and Motivation

2.1 Information Visualization

Many aspects of a visualization design are driven by the type of data we are looking at. In order to become understandable by users, data sources are often transformed along a process (the visualization pipeline) that transforms raw

[3] https://data.world/data-science-by-the-numbers.

data into a graphical representation that fits on the user screen. This process is shown at Fig. 1 encompasses four operations: (i) **data acquisition**; (ii) **filtering**; (iii) **visual mapping**; and, (iv) **rendering**. It is worthy of notice that user interaction (shown as dashed lines) might affect all operations in the pipeline.

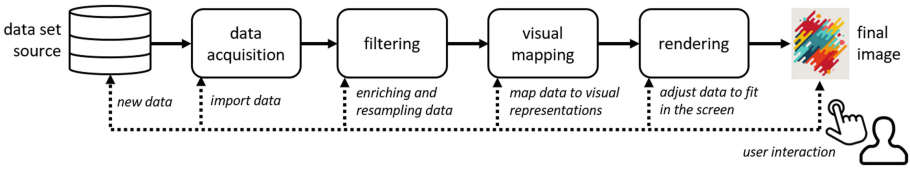


Fig. 1. Information visualization pipeline, adapted from Munzner (2014) [2].

Data acquisition is one of the most complex problems to be solved, mainly because the dataset might be available in various formats and most of visualizing techniques are specifically designed to handle a particular type of data (such as tables, clusters or lists). But there are exceptions such as Prefuse [8] which is an extensible user interface toolkit for crafting interactive visualizations that can handle both structured and unstructured data. And yet, the visualization process with Prefuse starts with abstract data represented in some canonical form (such as unstructured, graph, and tree data). In most cases, if the dataset is not on the format of the tool, it becomes tough to get benefits of the visualization technique.

When a dataset is loaded into the tool, it might contain information that is not relevant for solving the problem we are looking at. So that filtering operations come in place to remove noise, to fix attributes (ex. wrong character encoding), and to enrich the dataset (ex. add missing labels). An example of the use of filtering operation as a key feature of tools is illustrated by WebGIVI [9] which helps researchers to interpret large gene datasets by associating genes and informative terms (iTerm) that are obtained from the biomedical literature.

The visual mapping is at the core of the design of information visualization techniques. It allows the association of data attributes (such as gender and age) to visual variables (as form, color, size or texture). Mapping can be hard-coded or adjusted by the users. A good example is uVis Studio [10] which allows developers to compose visualizations by dragging-and-dropping building blocks, then binding controls to data and visualizing results with immediate feedback. Techniques such as dynamic bidding provide flexibility and interactivity for users to customize their views according to their needs.

The rendering operations define how the visualization techniques are displayed to the users. At this step, the tools might perform geometric transformations to make data to fit in the screen. The rendering also defines if the visualization is to be seen in a standard application or an element that can be integrated into another context of use. A flexible rendering is illustrated here by the framework Webcharts [11] which can adapt the rendering to three types

of users: for developer who uses the framework for creating an application; for the visualization developer, who extends the framework with new visualizations; and for the end-user, who may dynamically change the visualization of the data in the application, with no need of waiting for an update of the application that incorporates the latest visualizations.

The pipeline, shown in Fig. 1, is part of all visualization tools regardless of the technology used for the implementation. As far as Web technology is a concern, there are many libraries based on JavaScript that manipulate DOM and CSS to build visualization techniques that can be displayed inside the Web browser. A very well-known library is D3 [4] [12] which already offers a huge collection of interactive visualizations. The very common use of these libraries is to feed the visualizations with data that comes from some API or fixed data specified by Website developers. In most cases, the creation and the use of visualization techniques still remain something very technical that requires programming skills.

Viégas et al. [13] were pioneers in the democratization of information visualization techniques over the Web; their Web site called ManyEyes allowed people to create visualizations based on a predefined set of techniques available. Data acquisition in ManyEyes was simplified at the most, requiring a simple cut and paste; but it was not possible to connect tools for automating the information extraction from the Web. In addition to that, the rendering of the visualizations created by ManyEyes is not flexible and they cannot be integrated into other contexts of use than the Web site.

2.2 Web Scraping and Web Augmentation

We suggest that information visualization of datasets over the Web can be enhanced with Web augmentation and Web scraping technology.

Web Scraping allows transforming unstructured data available on the Web, typically in HTML format, into structured data that can be analyzed and stored analyzed in a central local database. For example, MeatBrain [14] is a tool that extracts data from Web sites and, eventually, aggregates different data into a new Web page. It is also very common that scrappers let their users define which part of Web sites to extract, meanwhile others may do it automatically.

Web scrapers are often the base for other applications such as search engines, Web automation, Web testing, and Web augmentation tools. It is interesting to notice that, although not every Web augmentation tool employs Web scraping, most of them contain some scrapper functionality that is used to parse the Web pages' DOMs in order to materialize the augmentation. Web augmentation typically allows to adapt existing third-party Web sites in order to add new content or functionality [15] and we suggest that it can be a suitable alternative to integrate visualization techniques into Web sites that lack visualization features.

There are different alternatives to achieve Web Augmentation at client-, server- or proxy-side. Client-side scripting is the most common alternative that can be evaluated through browser weavers like Greasemonkey[5]) or browser extensions.

[4] https://d3js.org/.
[5] https://www.greasespot.net/.

Annotation [16] is a broadly used technique to configure these underlying Web scrappers following a manual or semi-automatic approach. Actually, some Web augmentation approaches based on annotations arose to improve information visualization. For instance, Reform [17] allows developers to define general purpose applications that require some information to work. In this sense, end-users are responsible for the web content annotation from where that information must be extracted.

Other Web augmentation approaches may work based on an automatic scraper because their augmentation effect is not variable. VizMe [18] is a tool supporting an approach for handling additional data and tasks through augmented browsing. It is intended to provide extra information to the user in the same context of use, therefore, avoiding the switching between Web pages. That work emphasizes the visualization of such further data into the browsed Web page; they deal with the problem of how additional information is communicated to the user. They propose visualizations at different levels: visual cues at micro-level for hypermedia items and additional layers at macro-level for Web pages. At the micro-level, they present time-referenced Google data in a timeplot when the user highlights some Web content. At macro-level, they offer a wide range of visualizations on a floating panel, as a tag cloud based on the important words from the text on a Web page, a search engine to Google extra information or an editor to merge content from different pages. Similarly, another approach (Enhanced Web Page Content Visualization with Firefox) use natural language processing and machine learning techniques to help users to get a better overview of the pages they read, presenting graph-based visualizations.

3 Augmenting Web Sites with Visualization Techniques

In this paper, Web Augmentation (WA) is used to allow end users to build on-demand visualizations of semi-structured data sources available over the Web without changing the user's context of use, which means that visualizations are embedded into the web site users are visiting. The data acquisition is simplified by allowing users to select raw data presented in the Web page and turn them into visualization. This solution has the advantage of refreshing the visualization automatically when the Web page is updated. We also propose to reuse the data in search-results or with documents sharing the same structure. It is also possible to track the changes for a concrete element in the DOM through time, in order to analyze its evolution through visual means.

Providing users with a means to visualize any third-party semi-structured Web content presents some challenges from the point of view of the Web Engineering, mainly at the beginning of the visualization pipeline, where the data acquisition happens. The different structures inherent to the data representation in a page (HTML elements) must be understood to automatically extract and interpret their content to create a dataset serving as the input for a visualization. Moreover, first, it is mandatory to understand which are the HTML structures that may represent a target dataset to be visualized in a new way.

In this context, we formulated an initial set of questions: how many such HTML structures do exist? Can users benefit from visualizing existing data spread over the Web through alternative visualizations? Are augmentation-based-visualizations useful to solve any general-purpose task? Or is it better to use domain-specific ones? Does the user feel in control by using visualizations through WA or does he want more expressiveness power? Is specialized domain-knowledge a requirement for applying a visualization into any Web page? This work is a first step towards answering those questions, and Web Augmentation is presented as a bridge for joining Web Scraping and the benefits of visualization techniques for solving tasks without changing the context of use where data appears. It is not just about expanding the limits of technology but also enhancing the user experience in any Web page –even third-party– by the addition of a new feature. We aim at covering the gap between the existing Web scraping and visualization tools and techniques. In this approach:

1. users –with no need for knowledge in low-level scraping– can abstract raw data on a Web page into a data model specification (DMS),
2. users choose and apply alternative visualizations for the DMS
3. a repository of infovis augmentations do exist
4. developers can extend the existing visualizations in case existing specific visualizations do not cover a concrete task or domain, so users can apply them on any existing and third-party Web page

4 Web Sites: A Perspective on Content, Structure and Time

A first step towards the visualization of third-party raw data in the Web is to analyze how much and how diverse are the HTML structures presenting homogeneous content on the Web. To do so, we choose to analyze a sample of sites that can match into a table dataset type [2]. In this sense, our target datasets may be referred to as tables, which have rows (members), columns (variables) and cell values (datums).

To avoid sample bias, we took the list of sites from the top 50 popular sites according to Alexa's ranking for Argentina[6] as the target sample. We considered all the sites with a collection of at least five homogeneous elements representing a dataset member with more than a single variable. Regarding the dataset variables, only those that are present in all occurrences of the dataset members were considered. Besides, HTML elements not containing textual raw-data were not taken into account (e.g. images with no alternative text, «like» or «share» action buttons). The target datasets were searched in a limited part of the site: the homepages of the sites. If there was no data to visualize in the homepage (e.g., Google's default page) we triggered a search using the search engine of the site, to check if their Search Engine Results Pages (SERP) may contain items that may represent a dataset. The remaining pages of the sites were not

[6] https://www.alexa.com/topsites/countries/AR Dec. 18th, 2018 at 22:00 h UTC-3.

analyzed. If more than one possible dataset was detected for a site, we studied only the one with the most members and variables, respectively. The keywords used in the case of searching were «facts» and «certificado», respectively. It is worth mentioning that all the sites have been analyzed in a private-browsing tab, except for the ones that require log-in (e.g., Facebook or Twitter). From the 50 sites, we kept only 42 sites for analysis; we discarded 3 sites not meeting the requirement of content suitable for all audiences, 2 sites with the same domain, 2 sites that were offline at the time of analyzing and 1 site with a broken engine.

From the 42 sites, only 10 did not present any data with a heterogeneous structure. This leaves us with 76% of sites with data that may be visualized in an alternative way. These sites use different HTML elements to represent the data: 2 of them do it through a table («table»), 3 through an ordered list («ol»), 5 through an unordered list («ul»), 6 through a set of article elements («article»), and 16 through a hierarchy of homogeneous divs («div»). As shown in Table 1, we classified those cases in 3 categories: HTML tables, HTML lists and HTML hierarchical containers. In this work, we propose at least 3 kinds of dataset extractors.

Table 1. HTML structures

Dataset presentation	HTML Table	HTML list	HTML hierarchy
Dataset	table	ol/ul	div
Variables/columns	thead > tr	-	-
Members/rows	tbody > tr	li	div/article
Datum/cell	td > *	*	*
Occurrences in the sample	2	8	22

Regardless of such general HTML structure, different combinations of inner elements were found to present the datasets datum or table's cell value, which is the meaning of the «*» symbol in the Table 1. For instance, in YouTube's site we considered two anchors presenting the video title and video category, respectively, and two spans for the views and the date of publication. Since these data are shared by all the analyzed instances, they represent a variable of the dataset. The shared variables make it possible to claim that raw-data in existing and popular Web sites is comparable, and that it may be the target of our proposed visualizations. On average, the amount of dataset members analyzed ranged from 5 to 74, with an average of 18.6 and a standard deviation of 14.8 occurrences. We also found an average of 3.4 variables of the members, ranging from 2 to 6, and with a standard deviation of 1.2 variables.

We also observed that different mechanisms must be implemented for updating the visualization when it is required to include extra members from a dataset presented in a different context (e.g., on the second page of a dynamic table, or a SERP). Just 8 of the 32 sites with possible datasets had paginated members.

Therefore, in the 75% of the cases it was possible to retrieve extra members for the dataset through a user interaction: in 11 cases when the page is scrolled down, and in the remaining cases when a single link is clicked (10 cases, e.g., the «next» anchor), multiple links (2 cases, e.g. «page 1» or «page 2» anchors), or a button (1 case).

Obtaining such extra members (e.g., more videos appearing at the bottom of the page when the user scrolls down on the search engine of YouTube) or accessing a similar page which presents different data (e.g., two different pages of a YouTube video), allow reusing the same structure understanding. Such elements can be referenced by evaluating different Web locators [19], like XPath, CSS or JQuery selectors. We previously worked on the definition of user-defined scrappers capable of extracting similar elements loaded in different contexts [16]. So far, everything seems to be a question of how to map different HTML structures to the constitution of a dataset, and how to reuse the initial selectors to get more HTML elements to consider when the information is paginated. However, it is also a matter of time, since using the same selectors allows obtaining the same information at different times. Moreover, although a website may change over time (as you can check by using the Internet Archive [7]), Aldalur and Díaz [19] presented an approach for generating regenerative locators that use contingency data to evaluate alternative location strategies in case the DOM of a website changes. They validated their approach by taking a sample of a webpage from 8 websites every three months, and they find out that using their resilient locators, they were able to successfully regenerate the locators of the 73% of the samples. Therefore, using a locator over time for extracting elements and creating one or multiple datasets over time is plausible.

5 AlVis: An End-User Tool for Web Content Visualization

In the previous section, we presented the typical information visualization pipeline adapted to end-user activities in scenarios where they want to add alternative visualization to existing and third-party Web content. Moreover, we presented an analysis that we have made over several kinds of Web page's DOM structures in order to understand how to extract datasets from these semi-structured content. In this section, we present AlVis (ALternative VISualization through web augmentation), our end-user tool for visualizing Web content which is deployed as a Web extension [8]. We first present the use process of this tool, explaining the matching between interaction steps in the tool with the steps explained in the adapted pipeline. Later, we show the tool in action through some examples.

[7] https://archive.org/web/.

[8] AlVis prototype is publicly available https://github.com/gbosetti/alvis.

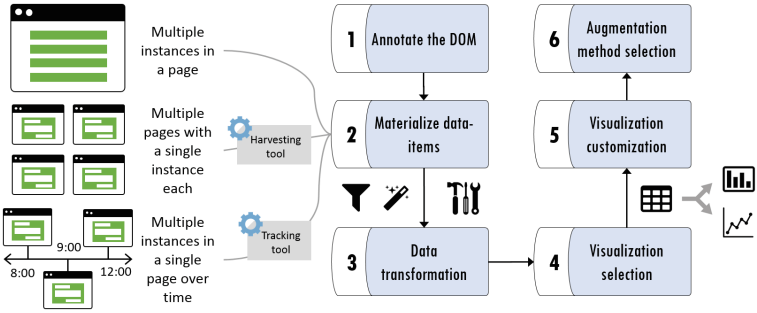


Fig. 2. AlVis use process for visualizing third-party raw-data in Web pages

The AlVis process (Fig. 2) requires six interaction steps by end-users:

1. DOM annotation (corresponds to the Data Acquisition step in the pipeline): define which part of the current Web site's DOM will be extracted. This step can be manual or semi-automatic. For a manual step, an annotation tool is required such as the one we have defined in previous work, called WOA (Web Objects Ambient) [16]. The semi-automatic way requires that users choose some semi-structured data automatically discovered by the AlVis extractors. For this regard, and based on the analysis presented in Sect. 4, we have defined the following extractors:
   - TableExtractor. Through this strategy, all the HTML tables present in the DOM are retrieved and analyzed. The extractor checks for the definition of the «thead» to identify the variables of the dataset, and the «tbody» to generate the output. In both cases, what is extracted are the children of such elements: a collection of «tr» elements. If multiple elements compose a «tr» element, these are split into new columns, in case the user needs to use them separately. The name of the column is the same but with the addition of an index.
   - ListExtractor. This strategy retrieves all the «ol» and «ul» elements from the current document. For each list, it takes all the «li» elements as possible members of the dataset. The variables are generated in a second round, by traversing all the possible members of the dataset and keeping just the leaf children elements that are present on all the members, based in its type. This means that if an «ol» has 5 «li», and only two of them have an element «anchor», then it is not considered as a variable. Under this strategy, the variables exist but their names are not representative: it is a combination of the name of the DOM element type concatenated with an index. The user will be able to redefine his name when manipulating the data after its extraction.
   - HierarchyExtractor. In this case, the variables are created in the same way as for the ListExtractor. What's different is the detection of similar elements representing the members of the dataset to be extracted (as the «li» elements inside a list for the ListExtractor). In this case,

the detection of similar elements is conducted by traversing the full body of the page's document looking for potential «container» elements that are not a «script» element, and that contain more than five children. Then, the children of each potential container are analyzed to check that more than the 50% of the elements are instances of the same type of element (e.g., articles, divs, ytd-video-renderer). If so, the instances of the predominant kind of element are considered as the members of the dataset to extract, and the variables are extracted in the same way as for the ListExtractor.

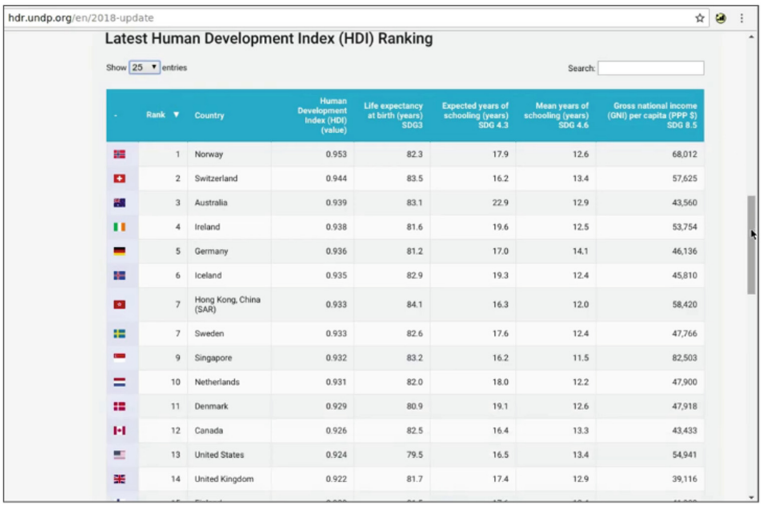
Both methods (manual and semi-automatic) generate a DOM annotation template that is stored in the AlVis local storage, in this way AlVis can be aware of the desired data structure for a particular Web site when this is loaded again in the future.

2. Data-Items materialization (corresponds to the Data Acquisition step in the pipeline): materialization is the process by which a DOM extractor parse the Web page's DOM to extract the data and their underlying data model. For any of the extractors defined for AlVis, the output is always a JSON that would be used for the further steps in this process.
3. Data transformation (corresponds to the Filtering step in the pipeline): some data could require to be curated by end-users before going on into the visualization steps. For instance, if some value extracted must be passed through a transformation function, or even if the data model requires some refinement, such as adding or changing naming columns heads. Moreover, this interaction steps allows users to delete data that is not required for the visualization.
4. Visualization selection (corresponds to the Visual Mapping step in the pipeline): This step allows end-user to choose a kind of visualization from the currently available ones. Although the AlVis tool includes a framework for adding new kinds of visualizations, the current prototype already covers the most common ones. It is important to mention that for the same dataset, several visualizations can be used.
5. Visualization customization (corresponds to the Visual Mapping step in the pipeline): Once a visualization is chosen, the user may customize which values to use, and other several aspects related inherent to the visualization being configured.
6. Augmentation method selection (corresponds to the Rendering step in the pipeline): finally, the last interaction steps in AlVis let users define how the alternative visualization must be rendered. Visualization could be added in a pop-up window or can be woven in the original Web page's DOM with different insertion strategies.

To illustrate this process, we present an example. For the sake of space, the example is based on using a semi-automatic extraction. The process starts with the user navigating the Web and identifying a raw dataset. For instance, the Latest Human Development Ranking by the United Nations Development Programme [9]. A screenshot of the page without augmentations can be found

[9] http://hdr.undp.org/en/2018-update.

in Fig. 3. In such page there is a table reporting variables as «the expected years of schooling» and «the gross national income» by country. Consider that a user wants to take his customized ranking as the dataset to visually identify the countries with higher gross national income and their proportion concerning the countries with the lower values. In the same Figure, it can be observed that the page is presenting 25 results by page. The user may want to create a visualization just with such a number of members, or he may want to do it with all the paginated results in the HTML table. For a matter of space, we will explain the simplest case.



-	Rank	Country	Human Development Index (HDI) (value)	Life expectancy at birth (years) SDG3	Expected years of schooling (years) SDG 4.3	Mean years of schooling (years) SDG 4.6	Gross national income (GNI) per capita (PPP $) SDG 8.5
	1	Norway	0.953	82.3	17.9	12.6	68,012
	2	Switzerland	0.944	83.5	16.2	13.4	57,625
	3	Australia	0.939	83.1	22.9	12.9	43,560
	4	Ireland	0.938	81.6	19.6	12.5	53,754
	5	Germany	0.936	81.2	17.0	14.1	46,136
	6	Iceland	0.935	82.9	19.3	12.4	45,810
	7	Hong Kong, China (SAR)	0.933	84.1	16.3	12.0	58,420
	7	Sweden	0.933	82.6	17.6	12.4	47,766
	9	Singapore	0.932	83.2	16.2	11.5	82,503
	10	Netherlands	0.931	82.0	18.0	12.2	47,900
	11	Denmark	0.929	80.9	19.1	12.6	47,918
	12	Canada	0.926	82.5	16.4	13.3	43,433
	13	United States	0.924	79.5	16.5	13.4	54,941
	14	United Kingdom	0.922	81.7	17.4	12.9	39,116

Fig. 3. A capture of the HTML table to extract data from

The first step is to extract the data to create the user's dataset of interest. It involves using one of three strategies, matching the structures mentioned in Table 1. All the extraction techniques generate the same output: a JSON with the variable names, and the members of the dataset. In case any element is not detected, it is defined as «undefined».

Under our approach, such extractors are evaluated when the user clicks the browser action of the AlVis extension (first step of Fig. 4); it is a button in the browser's toolbar that can be clicked to evaluate all the extractors with the content of the current page. From that moment on, an «extract» button is added at the bottom of all the DOM structures recognized by the extractors in the current webpage (step 2 of Fig. 4) without any extra user intervention, transparently performed.

When the user clicks the «extract» button, the extracted dataset is shown below in a new div under the HTML structure, which presents an editor to manipulate the dataset and use it through different visualization techniques.

This is the second step the user must carry out. For instance, she can change the variable names, remove variables or members, transpose the matrix, apply operators to the data. A screenshot of the extracted data presented through the editor is shown in the last step of Fig. 4.

The results of the two first steps are the transformation of raw data into a dataset. Both steps can be envisioned as part of what Card et al. [20] calls «Data Transformation» in their well-known model. What follows is the visualization selection and its customization.

Regarding visualization selection, our approach contemplates a framework where the visualizations represent an extension point. We provided a base of visualizations, like the ones listed at the top of Fig. 4, but these can be extended. The visualizations are presented according to the dataset characteristics. For instance, some of them require to have a mandatory variable name or are designed for a concrete kind of data: continuous, discrete or categorical. However, such a process is transparent for the end user, who needs to choose any available visualization and configure it if required. For instance, in Fig. 4 the user is required to choose two variables as the input for the X and Y axis of the bar chart. He can also use the control at the bottom to zoom in or zoom out part of the graph, including or excluding some bars on both sides of the graph.

A playlist with a video demonstration concerning this and other scenarios is also available online [10].

6 Validation

Before starting with the development of our prototype, we checked how much data available in the Web is a potential target to be visualized. Starting from the same sample described in Sect. 4, which was focused on the kinds of HTML structures, we also analyzed if such structures have data that makes sense to visualize without making data transformations. We discarded all the cases with no numerical variables and no repeated textual values or dates (e.g., a new's dataset with just two variables: «title» and «description»). The number of highlighted sites was 10, leaving 22 sites with data that could make sense for some user to visualize. In order to check the extractors, and for the sake of space, we took the two first sites from the sample matching each type of HTML structure (Table 1), and we used our described techniques to check if the proposed extraction techniques succeeded. The sites chosen were youtube.com, clarin.com.ar, wikipedia.org, twitter.com, blogger.com, and bna.com.ar. The extractors were successfully tested in the six sites, these were capable of extracting all the default members of the dataset and observed properties, with no exception.

[10] https://www.youtube.com/playlist?list=PLHuNJBFXxaLBFgtbBCZ7kOUUFd-Z3aaJK.

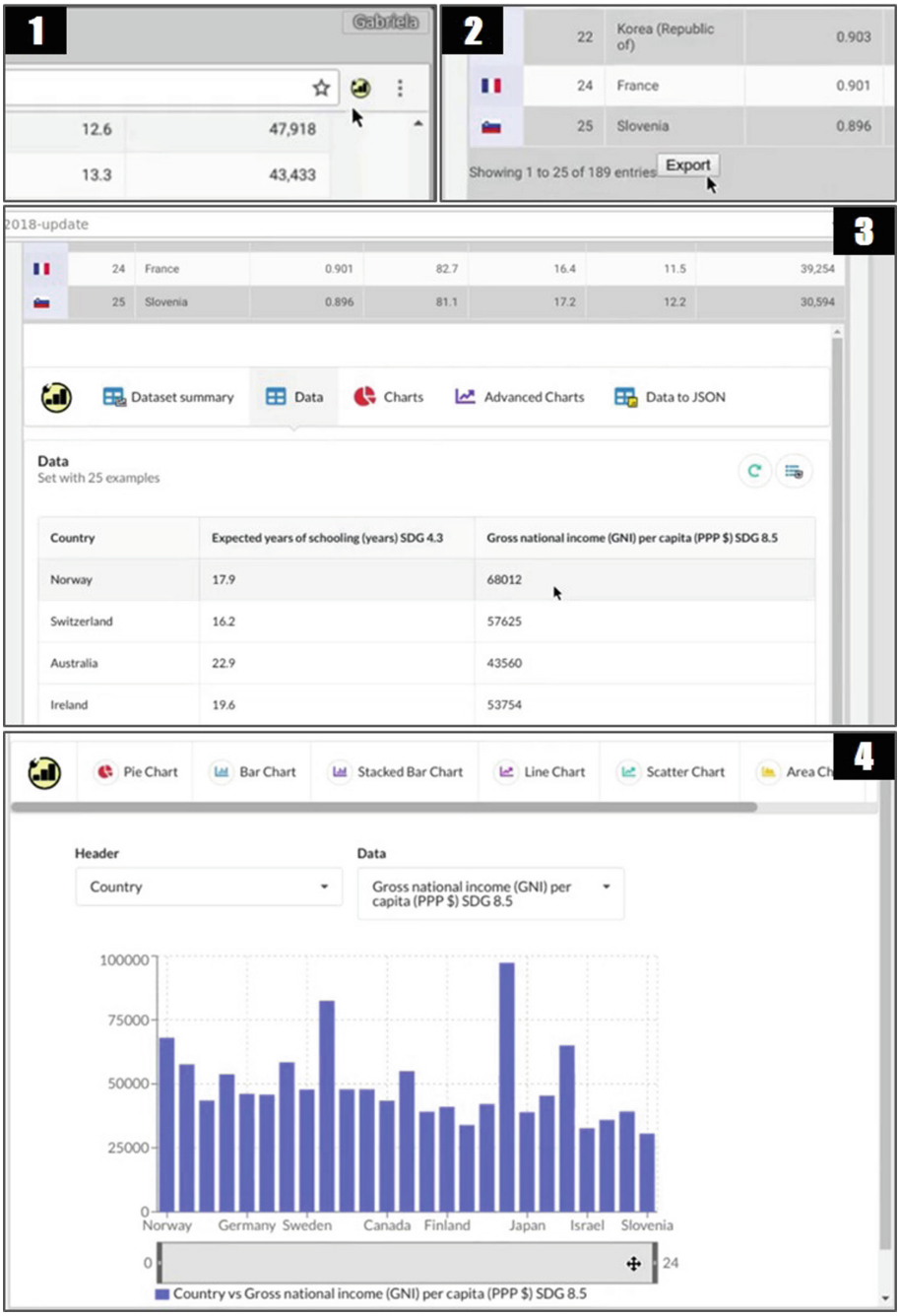


Fig. 4. Extracting and visualizing data from a page

7 Conclusions and Future Work

As the reader knows, the Web is a powerful platform for an extensive range of user activities, among which learning from the available information is not very well supported. As we know them, Web Browsers are a tool for browsing Web sites, but these hardly have evolved to support the complexity of available information, although there have been considerable advances in support of security issues as well as to support the technologies that have arisen during almost 30 years of Web applications evolution.

The kind of approach we present in this paper, and other compelling works in the context of Web mashups and Web Augmentation, aim to adapt Web browsers to reach new kinds of interactions that empower end-users to interact with Web content beyond the interactions that Web applications and Web Browsers allow. From our humble point of view, these approaches are vital given the advance of users capabilities that are very often not adequately addressed.

In this work, we analyze how information visualization would improve data consumption and use by end-users. We mainly analyze how the formal pipeline for information visualization should be adapted or applied by end-users (which are not necessarily experts in this area) for visualizing Web content. The engineering problem behind our approach is threefold: one problem is scraping semi-structured data from Web under demand into the Web browser to create datasets, a second problem is how to obtain visualizations from these datasets, and finally how to plug in-context new pervasive visualizations for any Web site.

At this moment, we are designing a user evaluation for our approach to process and visualize Web information. We firmly believe in the idea that this kind of browser's behavior would make the user's tasks faster and also the user's understanding of information deeper. Moreover, we plan a more in-depth analysis of multiple websites to understand how much content share a similar structure and find which are the best alternatives for generating proper selectors. At the same time, we are studying how information can be usefully extracted to help users in its analysis and use in other ways. For instance, for creating dynamic datasets using temporal information that Web sites change with a specific frequency. Furthermore, using user-driven annotations for extracting complex instances of information that are composed without a semi-structured presentation enabling automatic extraction.

Finally, other more technological aspects for making easier the use of this tools are devised, such as repositories for quickly sharing and maintaining visualizations and the implementation of a library that allows Web developers to apply this kind of visualizations internally.

References

1. Sanou, B.: Measuring the information society report 2018. In: International Telecommunication Union, Geneva, Switzerland (2018)
2. Munzner, T.: Visualization Analysis and Design. AK Peters/CRC Press, New York (2014)

3. Yi, J.S., ah Kang, Y., Stasko, J.: Toward a deeper understanding of the role of interaction in information visualization. IEEE Trans. Vis. Comput. Graph. **13**(6), 1224–1231 (2007)
4. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. Craft Inf. Vis., 364–371 (2003)
5. Price, M., Crumley-Branyon, J., Leidheiser, W., Pak, R.: Effects of information visualization on older adults' decision-making performance in a medicare plan selection task: a comparative usability study. JMIR Hum. Fact. **3**(1), (2016)
6. Mahendiran, J., Kirstie Hawkey, N.Z.H.: Exploring the need for visualizations in system administration tools. In: CHI 2014 Extended Abstracts on Human Factors in Computing Systems, pp. 1429–1434. ACM (2014)
7. de Borja, F.G., Freitas, C.M.D.S.: CivisAnalysis: interactive visualization for exploring roll call data and representatives' voting behaviour. In: 28th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI 2015), pp. 257–264. IEEE Computer Society (2015)
8. Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 421–430. ACM (2015)
9. Sun, L., et al.: WebGIVI: a web-based gene enrichment analysis and visualization tool. BMC Bioinf. **18**(1), 237 (2017)
10. Pantazos, K., Kuhail, M., Lauesen, S., Xu, S.: uVis Studio: an integrated development environment for visualization. Vis. Data Anal. **2013**, 8654 (2013)
11. Fisher, D., Drucker, S., Fernandez, R., Ruble, S.: Visualizations everywhere: a multiplatform infrastructure for linked visualizations. IEEE Trans. Vis. Comput. Graph. **16**(6), 1157–1163 (2010)
12. Bostock, M., Ogievetsky, V., Heer, J.: D3 data-driven documents. IEEE Trans. Vis. Comput. Graph. **17**(12), 2301–2309 (2011)
13. Viégas, F.B., Wattenberg, M., van Ham, F., Kriss, J., McKeon, M.M.: ManyEyes: a site for visualization at internet scale. IEEE Trans. Vis. Comput. Graph **13**(6) (2007)
14. Teixeira, J., Barata, G., Gonçalves, D.: Metabrain: web information extraction and visualization (2012)
15. Díaz, O., Arellano, C.: The augmented web: rationales, opportunities, and challenges on browser-side transcoding. ACM Trans. Web **9**(2) (2015)
16. Firmenich, S., Bosetti, G., Rossi, G., Winckler, M., Barbieri, T.: Abstracting and structuring web contents for supporting personal web experiences. In: Bozzon, A., Cudre-Maroux, P., Pautasso, C. (eds.) ICWE 2016. LNCS, vol. 9671, pp. 77–95. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38791-8_5
17. Toomim, M., Drucker, S.M., Dontcheva, M., Rahimi, A., Thomson, B., Landay, J.A.: Attaching UI enhancements to websites with end users. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1859–1868. ACM (2009)
18. Nguyen, D.Q., Schumann, H.: Visualization to support augmented web browsing. In: International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), pp. 535–541. IEEE/WIC/ACM (2013)
19. Aldalur, I., Diaz, O.: Addressing web locator fragility: a case for browser extensions. In: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 45–50. ACM (2017)
20. Card, S.K., Mackinlay, J.D., Shneiderman, B.: Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, San Francisco (1999)