AUTOMATIC HTML FORM FILLING ASSISTANT

*Mariya Zhekova *, Nedyalko Katrandzhiev*

Department of Computer Systems and Technologies,
University of Food Technology, Plovdiv
Bulgaria

* Corresponding Author, e-mail: m_jekova@uft-plovdiv.bg

Abstract: Web application users fill data in web forms every day, and problem situations often arise due to various reasons (misunderstanding the requirements, entering wrong data or value outside a certain range, or applying some regular expression to a certain field, etc.). The paper proposes a hybrid approach to identify and classify different elements in a web document in an accurate and efficient way that optimizes and automates the process of data filling in web forms by users. Machine learning techniques, heuristics and dynamic analysis of HTML objects were used, and a script was developed to overcome difficulties in filling out forms. The advantage of the proposed script is that it can work with HTML documents regardless of the domain area, specificity and nature of their content, saving time in addition. The characteristics of the attributes and constraints of the input elements in web forms are extracted fully automatically and the generated messages to the users are synthesized based on a generalized structural rule.

Key words: web recognizing, web content extraction, HTML forms detection

1. INTRODUCTION

One of the main characteristics of the digital age, in addition to contemporary intelligent systems [1] and network technologies [2], is the continuous development of user communications and remote access to information resources. It is known that web pages (as part of more complex web systems, platforms and applications) are the main means of information dissemination on the Internet. In essence, they contain recognizable elements and constants that together describe the specifics and content of the documents they construct.

When accessing and working with more complex information systems, users are often required to fill in data in web forms to collect user information, which can cause undesirable situations. In addition, the collection of personal data must comply with the privacy requirements clearly defined in the European document “General Data Protection Regulation” (GDPR) [3]. In addition to the stated legal regulation of data, situations may often arise that will make it difficult for certain users without the necessary experience in network processes, for example, due to not having a good knowledge of the procedures, entering wrong data, etc.

To solve such problems, the article proposes a hybrid approach for identifying and classifying the components in a web document, allowing for efficient organization and optimization of the process of data filling in web forms. HTML content extraction accesses DOM (Document Object Model) objects and, through machine learning techniques, identifies web components using dynamic analysis of HTML structures and calls, their attributes, values, and properties.

The technique is a type of web scraping (crawling) of an HTML document. Web scraping is a technology that allows us to extract structured data from text such as HTML. Using web scraping to collect data allows us to collect prices from retail sites and provide additional details, web scraping can also be used to gather intelligence on illegal businesses to provide law enforcement valuable data that would not be available through conventional methods. Data acquisition is the first phase in any scientific investigation; this is the step where data obtained from sources such as personal data, company sales records and financial statements, or from public sources such as journals, websites and open data. Website analysis, crawling and data organization are the three main, intertwined processes of online scanning [4].

The aim here is not to make predicative analyzes and inferences about malicious code, nor validation checks for a web form, existence of cyber-attacks on the website, logical errors, quality or good practices related to the interaction between a website and its users. Our goal is to facilitate and automate the process of filling out HTML forms with user data, which is achieved by optimizing the design of web pages to collect information, embedding dynamic functions for reading HTML code, validating form data, reacting to events in the browser.

The proposed approach facilitates the filling of HTML forms by non-specialist users and automatically suggests to them what data they are expected to fill in specific input elements and is the basis of the so-called automatic web form filling assistant. This paper discusses the web scraping method, implementation and stages, and how it can be done through JavaScript code.

2. RELATED WORK

Many websites, especially those that dynamically generate HTML pages, contain explicit and implicit (hidden) structure, both in layout and content that can be used for information retrieval purposes. There are various web scraping techniques including traditional copy-paste, text grapping and regular expression matching, HTTP programming, HTML parsing, DOM parsing, web scraping software, vertical aggregation platforms, semantic annotation recognizing and computer vision web page analyzers [5].

Some of the automatic approaches to web element recognition and information extraction in the past rely on the DOM, others on regularities in HTML tags and attributes. The use of JavaScript and dynamic DOM manipulation on the client side of web applications is becoming a widespread approach to achieve rich interactivity and responsiveness in modern web applications [6].

Tools exist to extract information from web pages created using machine learning techniques [7]. Despite the inherent difficulty, there are general principles and algorithms that can be used to automatically extract data from structured web sites. An approach for automatic web content extraction and segmentation is proposed in [7]. Its methods do not require user input, but rely entirely on the layout and content of the web source. Two algorithms are described in the article. The authors show how each approach can exploit web site structure in a general, domain-independent way and demonstrate the performance of each algorithm on a set of twelve web sites.

There are software APIs that crawl web documents (sites, applications) and extract content from them. They operate on the principle of a full-fledged web browser (Chrome, Opera, Mozilla) or on the principle of the low-level hypertext transfer protocol (HTTP). API can be access for variety of different data types. Instead of manually copying and pasting information from the website and collecting it in electronic repositories, the content extraction service offers this functionality in applications, more precisely, automatically and faster than a human [4].

Availability of powerful client-side web-browsers, as well as the wide adaptation to technologies such as HTML5 and AJAX, gave birth to a new pattern in designing web applications called Rich Internet Application (RIA). RIAs move part of the computation from the server to the client. This new pattern of designing web applications led to complex client side applications that increased the interactivity of the web application [8].

There is a query-based tool for crawling of a web document that uses a set of keywords appropriate to the domain. This helps the tool get the most relevant links from the domain without actually going deep into that domain. In the proposed tool, the keyword list is passed to the search query interfaces found on the websites. The keyword query based focused crawler guides the crawling process using metadata. The keyword data set is used for creating effective queries and the result obtained are feedback to the system [9].

During the exploration of automatic dynamic web crawlers, were discovered two inevitable situations which have not been processed properly by existing tools. One is that some websites adopt CSS pseudo-class to locate clickable elements, which prevents the existing technologies from simulating user actions. Another is that pop-up windows can be triggered during automatically elements clicking, where the existing web crawler procedures will be terminated [10].

Finally, we can't depend on pre-built off-the-shelf tools to get the data we need, so we need to implement data extraction strategies, including specific attributes from web documents, to ensure that the data we populate, match what the business needs.

3. METHODOLOGY FOR OPTIMIZING THE PROCESS OF FILLING OUT HTML FORMS

Many websites and applications have forms to collect user data. In most computer information systems, communication with the user takes place through a user-friendly graphical user interface (GUI) [11]. The entry point or data source for the reasoning, algorithm, and programming implementation of the automatic assistant is an HTML form that builds the GUI (front-end code) of a web page.

To create a tool for assisted filling of web forms (regardless of the nature, content and software), the cross-platform **JavaScript** language was chosen to interact with the full potential of the HTML content of a given web page with input elements, facilitating their filling by user side. The **tooltip** concept was chosen, with the help of which instructions are shown to the user in the form of additional information for filling in a specifically selected element in a given web form. When a user holds the mouse pointer over an element (hover, focus, select effect), an instruction is displayed for its correct filling or what type of data is expected to be filled, in what limits it should be, is the field mandatory, does it follow a certain template for filling etc.

The process takes place in two stages.

- The first stage is the pre-processing of the document's DOM tree to obtain its full-text content and involves training its further behavior.
- The second stage takes care of extracting information and performing key activities for the purpose of the processes. In this particular case, the aim is to assist in filling out user data in HTML forms.

The proposed approach scans the DOM tree, finds attributes of input elements that trigger popup messages with hints to users

3.1. Automatic assistant for filling out HTML forms

Stage I– training stage

In the pre-training stage, the tool first processes the existing HTML content. The processing includes the removal of redundant and duplicate elements.

Redundant are those elements that are not essential for the purposes of the study (<br>, <hr>, &nbsp; <b>, <i>, <sup>, etc.). Whether text is bold or italic does not matter to the app's functionality. Input elements are the focus of the reasoning, and only for them are created tooltip instructions for filling in by users.

Inferring an input element type once ensures that the rule is applied every time the same element type is encountered again in the DOM tree.

Hyper Text Markup Language Document Object Model (HTML DOM) is a standard for getting, changing, adding, or deleting HTML elements. DOM performance is by defining objects and properties of all HTML elements, with methods to access them. With DOM, JavaScript can access all elements in an HTML document. All HTML elements are treated as objects [12].

Consider the following code fragment (Fig. 1):

```
<form>
<fieldset style="background:#e1eff2;">
       <legend><b>General information</b></legend>

       <label for="firstname">First name *</label>
         <input type="text" name="firstname" min="2" max="15" required/> <br>

       <label for="lastname">Last name *</label>
       <input type="text" name="lastname" min="3" max="20" required /> <br>

       <label for="username">Username *</label>
       <input type="text" name="username" required /> <br>

       <label for="pass">Password *</label>
       <input type="password" name="pass" required /> <br>

       <label for="sex">Sex</label>
       <input type="radio" name="sex" value="man" />
         <label for="man">Man</label>
       <input type="radio" name="sex" value="woman" />
         <label for="woman">Woman</label>

       <label for="age">Age</label>
       <input type="number" name="age" min="1" max="120" /> <br>

       <input type="submit" value="Save" />
         <input type="button" value="Reset" />
</fieldset>
</form>
```

Fig. 1. Fragment of code building the user interface of an HTML form

A sequence is a set of elements that are part of a web form for entering user data (Fig. 1). Multiple label and input elements are observed in the fieldset. Only the input fields fall within the scope of our reasoning. It is for them that the algorithm is proposed and the tool developed, as a result of which users will be able to receive instructions for filling out the web form (Fig. 2).

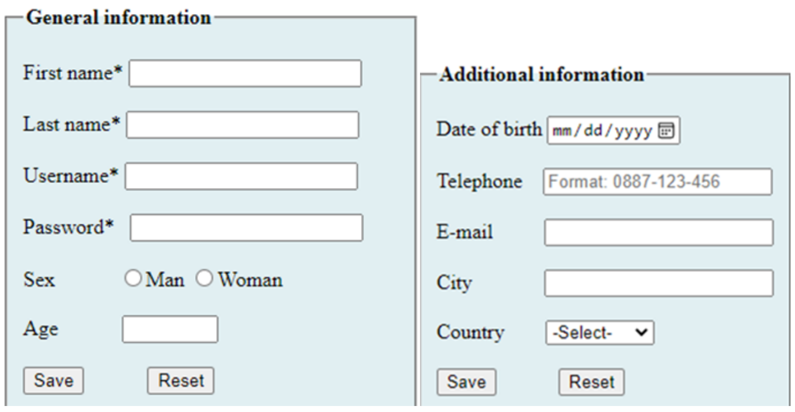


Fig. 2. Part of an HTML form for filling out user information

The result of key web information extraction is a list of elements or a collection of attributes (in the form of additional information) for each of the elements in the page's DOM (Fig. 2.).

In the example for all fields of type <input type=”text”> the logic will follow the same line, namely the user is instructed to enter some text message, for all fields of type <input type=”number”> the instructions will be filling in numerical data, etc. If there are attributes indicating the mandatory nature of the field, the string – "The field is mandatory for recording" is added to the indication. If there is a range for the numerical values min=”0” max=”150”, the message will be generated with valid values for the validity interval.

The **JavaScript scripting language** is used to form the rules, as it can interact both with objects from the DOM (Document Object Model) tree of the web document and CSS properties of these objects, as well as with plugins in the browser, data server, etc.

The DOM model is platform and programming language neutral and allows programming scripts to dynamically access and update the content, structure, and style of web documents. It defines the logical structure of the document and how it can be accessed and manipulated. The DOM model covers not only the structure of a web document, but also its behavior and the objects it consists of. As an object model, DOM identifies [13]:

- Objects – for presenting and manipulating a document;
- Semantics of these objects - study of their behavior and attributes;
- Connections and relationships between objects.

The **JSON format** for data exchange (JavaScript Object Notation) is also used, because of its speed, compatibility and independence. With its help, the necessary information is accessed and some of the processing in the HTML document is simplified, such as obtaining the names of the objects and the values of their attributes.

Directions are displayed in the GUI as tooltip messages that could be styled for a friendlier look. They can be "linked" to a specific element by an arrow, fade in gradually (kind of animation) or visualized as a modal window (modal box). We've chosen to show them to the right of the fields with a directional arrow.

It only needs to read a dataset from the DOMStringMap, where there is an entry for each input element property in the web document. Event handlers receive an object that contains specific properties and methods.

```
var document_info = document.documentElement.innerHTML;
```

The code above shows how the content of a web document can be written to a document_info string variable. In the resulting string, it is possible to search for the presence of different types of input fields and to generate tooltip messages that are displayed when positioned on the input elements and that will be helpful when filling in by users.

Stage II – extraction of DOM objects and values of their characteristics

In the detection stage, the preprocessing of input elements from the DOM tree and extraction of values of specific characteristics of them (eg type, required, etc.) is performed and the results are found in the same training stage. The type of the input elements is derived according to how the model was built and trained in the training stage. The decisions about the tooltip messages that accompany the work of the automatic assistant are also made based on the justifications in the training stage.

For easier and more intuitive management of web documents, we use the JavaScript library jQuery. It detects the ready state of the loaded HTML document via $(document).ready() and only then executes the JavaScript code. JQuery provides methods for many natural events that occur in the browser, such as click(), focus(), blur(), change(), on(), etc. A DOM element can be turned into a jQuery object to which jQuery methods are applied, using $(this) e.g. var label = $(this). In addition to the event object, the event handler also has access to the DOM element to which the handler was bound via the *this* keyword.

The code below represents (Fig. 3) part of the preprocessing of the object model of the considered example to obtain its full-text content and includes functions for extracting information about input elements and values of specific characteristics of them.

```
     function getAllParaElems() {
       var inputFields = document.genInfo.getElementsByTagName('input');
       var len = inputFields.length;
       console.log("There are " + len + " input fields.");

       var i,j;
       for (i = 0; i < inputFields.length; i++) {
             var attrInElement = inputFields[i].attributes;
             console.log(attrInElement.length);
             for(j = 0; j < attrInElement.length; j++) {
             console.log(inputFields[i].attributes[j].name        +"        -        "+
inputFields[i].attributes[j].value);
             if(inputFields[i].attributes[j].value === "text" && ) {
                    console.log("Expecting text information.");
                    if(inputFields[i].attributes[j].value === "min") {
                          const val = inputFields[i].attributes[j].value;
                          console.log("Expecting text information with minimum " + val +
" chars.");
                          }      }
     .............
             }      }
     ........... }
```

Fig. 3. A function that retrieves input elements and values of their characteristics

The snippet below demonstrates CSS code that is used to style the tooltip messages to users (Fig. 4) and demonstrates how to add an arrow to the bottom of a tooltip:

```
/* Tooltip text */
  .tooltip .tooltiptext {
  visibility: hidden;
  width: 300px;
  background-color: #555;
  color: #fff;
  text-align: center;
  padding: 5px 0;
  border-radius: 6px;

/* Position the tooltip */
  position: absolute;
  z-index: 1;
  bottom: 150%;
  left: 50%;
  margin-left: -60px;

/* Fade in tooltip */
  opacity: 0;
  transition: opacity 0.3s;
/* Tooltip container */
  .tooltip {
  position: relative;
  display: inline-block;
  }
/* Position to the bottom */
  .tooltip .tooltiptext::after {
  content: "";
  position: absolute;
  top: 100%;
  left: 50%;
  border-width: 5px;
  border-style: solid;
  border-color:  black  transparent
transparent transparent;
  }
```

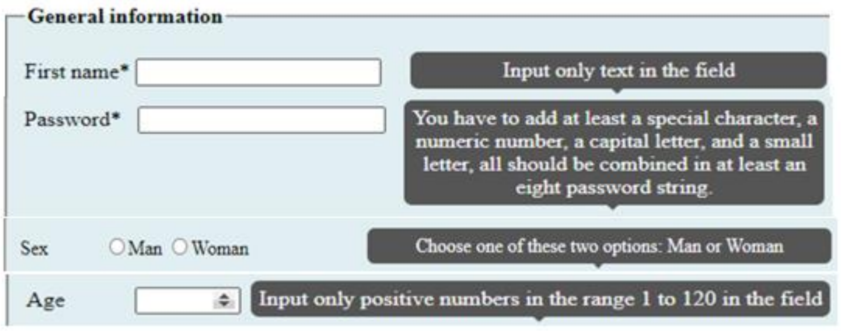
Fig. 4. CSS code that styles the tooltip messages

4. RESULTS

To create an arrow that should appear on a specific side of the tooltip, content: "" is added after the tooltip, with the after pseudo-element class. The arrow itself is created using border. This will make the tooltip look like a content balloon.

Positioned arrow with parameter top: 100% places the arrow at the bottom of the tooltip, left: 50% centers the arrow. The border-width property defines the size of the arrow, and margin-left the position of the arrow. The value of margin-left depends on whether the arrow should be centered. The border-color property is used to style the arrow.

Some of the text to generate the pointer messages is derived from attribute values in the web document's HTML code, e.g. the tokens Man or Woman in the string are taken from the value="Man" and value="Woman" attributes of the <input type="radio"> element in the HTML document (Fig. 5).



General information

First name* Input only text in the field

Password* You have to add at least a special character, a numeric number, a capital letter, and a small letter, all should be combined in at least an eight password string.

Sex ○Man ○Woman Choose one of these two options: Man or Woman

Age Input only positive numbers in the range 1 to 120 in the field

Fig. 5. Fragments of the same HTML form with user prompts generated

In the event that there are validating attributes specifying value arrangements, a time period, or a regular expression, the script is trained and forms a message to the client that includes the listed parameters. Thus, the user will not enter invalid values and waste time editing already entered information. JavaScript code seeks to automate communication between the client and the web page by implementing interaction functions through DOM objects to control their behavior in the browser.

5. CONCLUSION

The tool developed as a result of this study serves as an automatic assistant for filling out HTML forms for users to fill in data and supports reasoning about the interaction process between users and web applications, without the iterations affecting the servers. The reasoning also uses cleverly combined heuristic rules and machine learning techniques.

The script works on HTML documents regardless of the domain area, specificity and nature of their content. It does not know the scheme by which web content is organized in an HTML document, is not limited to a linear form, and can handle nested structures, which in most cases are HTML forms.

Using the script reduces user stress and helps the process of filling out data in web forms, saving time. The solution is independent of the domain area, fully automatic and does not require user intervention. Characteristics for attributes and constraints on input elements in web forms are extracted fully automatically and generated messages to users are synthesized based on generalized structural rules.

REFERENCES

[1] Kasabov, N. From Multilayer Perceptrons and neuro-fuzzy systems to deep learning machines: Which method to use? – A survey. *International Journal on Information Technologies and Security*, vol. 9, no. 2, 2017, pp. 3-24.

[2] Kravets, O.Ja., Atlasov, I.V., Aksenov, I.A., Molchan, A.S., Frantsisko, O.Yu., Rahman, P.A. Increasing efficiency of routing in transient modes of computer network operation. *International Journal on Information Technologies and Security*, vol. 13, no. 2, 2021, pp. 3-14.

[3] Tzolov, Tz. Data model in the context of the General Data Protection Regulation. *International Journal on Information Technologies and Securit*y, vol. 9, no. 3, 2017, pp. 113-122.

[4] Ahmad, K.M. Web Scraping or Web Crawling: State of art, techniques, approaches and application. *International Journal of Advances in Soft Computing & Its Applications*, vol. 13, no. 3, 2021, pp. 145-168.

[5] De S Sirisuriya, S. C. M. A Comparative Study on Web Scraping, in *Proceedings of 8th International Research Conference of KDU*. General Sir John Kotelawala Defence University, 2015, pp. 135–140, http://ir.kdu.ac.lk/handle/345/1051

[6] Mesbah A., van Deursen A., Lenselink S. 2012. Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes. *ACM Trans*. Web 6, 1, Article 3, March 2012, 30 p. https://doi.org/10.1145/2109205.2109208.

[7] Lerman, K., Getoor, L., Minton, S., Knoblock, C. Using the structure of Web sites for automatic segmentation of tables. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data (SIGMOD '04)*. 2004. New York, NY, USA, pp. 119–130. https://doi.org/10.1145/1007568.1007584.

[8] Taheri, M., Mohammad, S. Distributed Crawling of Rich Internet Applications. *PhD thesis*. University of Ottawa, Canada, 2015.

[9] Kumar M., Bindal A., Gautam R., Bhatia R. Keyword query based focused Web crawler, *Procedia Computer Science,* vol. 125, 2018, pp. 584-590, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.12.075.

[10] Li, Y., Peiyi, H., Chuanyi, L., Binxing, F. Automatically Crawling Dynamic Web Applications via Proxy-Based JavaScript Injection and Runtime Analysis. *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, 2018, pp. 242-249.

[11] Zhekova, M., Totkov, G. Model of process and model of NLP system, *IOP Conf. Series: Materials Science and Engineering*, vol. 878, art. 012028, 2020, IOP Publishing, doi:10.1088/1757-899X/878/1/012028.

[12] URL: https://www.w3.org/

[13] Gunawan, R., Rahmatulloh, A., Darmawan, I., Firdaus, F. Comparison of web scraping techniques: regular expression, HTML DOM and Xpath. *2018 International Conference on Industrial Enterprise and System Engineering (ICoIESE 2018)*, 2018, pp. 283-287. Atlantis Press, 2019.

Information about the authors:

Mariya Zhekova – Assistant professor and doctor in Informatics who currently works in the Department of Computer Systems and Technologies in University Of Food Technology - Plovdiv. Research programming languages, Databases and Artificial Intelligence related to natural language comprehension. Developing a model in which syntactic information and operations that were previously thought to be related to lexical entries are replaced by a syntactic-semantic functional structure.

Nedyalko Katrandzhiev – Professor in the Department of Computer Systems and Technologies in University Of Food Technology - Plovdiv. He interested in: - WEB programming; - Computer Graphics and Design; - Nondestructive Optical Methods for Fruit Quality Assessment; - "Seeing Through Layers" (STL); - Automatic Machines for Quality Assortment of Fruit Products; - Fruit Ripeness Determination by Electronic Sensing of Aromatic Volatiles; - Intelligent Systems; - High Speed Machines for Vision Inspection; - Sensory Systems for an Electronic Nose; - Science and Research (e-mail: ned_uht@abv.bg)

Manuscript received on 29 September 2022