applied sciences

Article

Tool for Parsing Important Data from Web Pages

Martina Radilova , Patrik Kamencay *, Robert Hudec , Miroslav Benco and Roman Radil

Faculty of Electrical Engineering and Information Technology, University of Zilina, Univerzitna 8215/1, 01026 Zilina, Slovakia
* Correspondence: patrik.kamencay@uniza.sk

Abstract: This paper discusses the tool for the main text and image extraction (extracting and parsing the important data) from a web document. This paper describes our proposed algorithm based on the Document Object Model (DOM) and natural language processing (NLP) techniques and other approaches for extracting information from web pages using various classification techniques such as support vector machine, decision tree techniques, naive Bayes, and K-nearest neighbor. The main aim of the developed algorithm was to identify and extract the main block of a web document that contains the text of the article and the relevant images. The algorithm on a sample of 45 web documents of different types was applied. In addition, the issue of web pages, from the structure of the document to the use of the Document Object Model (DOM) for their processing, was analyzed. The Document Object Model was used to load and navigation of the document. It also plays an important role in the correct identification of the main block of web documents. The paper also discusses the levels of natural language. These methods of automatic natural language processing help to identify the main block of the web document. In this way, the all-textual parts and images from the main content of the web document were extracted. The experimental results show that our method achieved a final classification accuracy of 88.18%.

Keywords: Document Object Model; extraction; world wide web

check for updates

Citation: Radilova, M.; Kamencay, P.; Hudec, R.; Benco, M.; Radil, R. Tool for Parsing Important Data from Web Pages. *Appl. Sci.* **2022**, *12*, 12031. https://doi.org/10.3390/app122312031

Academic Editors: Gaetano Zizzo and Alexander N. Pisarchik

Received: 18 October 2022
Accepted: 21 November 2022
Published: 24 November 2022

1. Introduction

Websites contain a huge amount of information such as the main text, offers, advertisements, banners, footers, sitemaps, title, summary, price, description, etc. With the increase in unnecessary content on the website it has become essential to eliminate unnecessary content and extract the needed content that can be used in word processing applications such as search engines, question answering systems, referral systems, trend detection/monitoring, assortment analysis as well as e-commerce market monitoring. Several studies [1,2] address this issue. They focus on the task of automatic determination of data/pattern extraction. However, these studies do not consider the time efficiency of this process.

On the other hand, there are also studies that focus on the acceleration of the extraction process, such as the decision on the extraction pattern [3]. Based on algorithms of the main data extraction from the site, other methods have been proposed. A suitable example is a method for data extraction based on rendering information and an n-gram model (DERIM) [4]. It is an automated method that can identify the main data area on the Hypertext Markup Language (HTML) page, but also provides the search result records (SRR) and attributes. Furthermore, the method analyzes the Document Object Model (DOM) tree and provides information related to its elements. In addition, DERIM can detect various recording structures.

As another mean of information obtaining, UzunExt could be mentioned. The UzunExt approach consists of two main components: crawling web pages and extracting data from them. It uses the additional information obtained from web pages during the crawling process for increasing the extraction time efficiency [2].

In addition to the presented state-of-the-art studies, there are many more, which could be taken into consideration. First studies of data extraction from a website include methods based on wrapper induction [5–8], but there are later ones, and these are already based on machine learning techniques [9–12].

This paper is an extended version of a paper published at the 2020 43rd International Conference on Telecommunications and Signal Processing (TSP) [13]. The paper [13] presented a tool for mobile phones, which also serves to parse the main content of HTML pages. However, it is created for android version 4 and less and is intended only for HTML news pages. Therefore, in this article authors extend the application from [13] to the version for android 10 and of course it has been also extended to HTML pages of different scope than just news. Subsequently, in further continuations, authors will expand it to machine learning, because the creators of web pages are advancing and introducing new techniques into them. Unlike the article that was published at the TSP conference, this contribution is supplemented with NLP technique. This tool checks every tag in a web document. When it finds using NLP technique a tag that contains any text, it determines whether it contains more than four sentences (min. five sentences). If it contains more than four sentences, we extract it from the web document as the desired content (the main content of the web document).

2. Document Object Model

Programmers and web designers have tested various ways how to access and manipulate the content of web documents. Going through the significant number of websites, it has been found, that almost all use different scripts to interact with visitors, whether for check forms or a pop-up drop-down menu. However, the basic model remains the same; the script accesses and manipulates the related parts of the document and the visitor see the result of this manipulation [14].

The main problem for the widespread use of scripting on the web has been non-standardized scripting interfaces. For designers, this meant, among other things, customizing documents for different web browsers so that the visitor would not have any restrictions during viewing or interaction of any restrictions. The World Wide Web Consortium (W3C), therefore, created the application interface Document Object Model, which solves these problems [14].

2.1. Markup Language Interface Document Object Model

The Document Object Model (DOM) is an application for programming interfaces for HTML and eXtensible Markup Language (XML) documents. It defines the logical structure of a document and the ways in which the document is accessed and manipulated [15]. The DOM interface allows programmers to create documents, navigate through them, and add, edit, or delete individual elements and content. The interface is platform-independent and thus can be used in a wide range of programming languages, development environments and applications [15]. The working group, which created the standardized Document Object Model interface, dealt with three markup languages: Standard Generalized Markup Language (SGML), HTML, and XML.

2.2. DOM Interface Definition

DOM interface definition in the W3C recommendation describes how the objects found in HTML or XML documents are manipulated. All other parts of the document are derived from the main interface—Nodes. Figure 1 shows the most used interfaces derived from this main node.

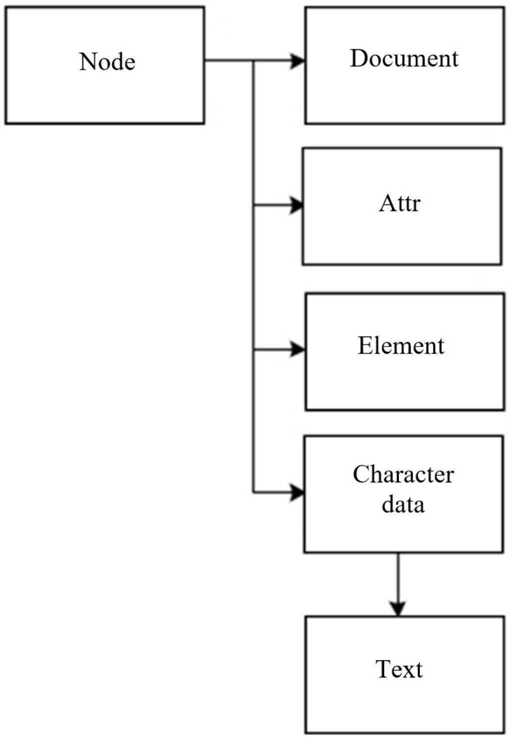


Figure 1. The structure of DOM inheritance [14].

Interface "node" is the main data type of the whole DOM, which represents a single node in the document tree. It contains methods for inserting, deleting, and replacing nodes, which are also available for all inherited interfaces [14,15].

Interface "Element" represents an individual component (element) of the document. It is derived from the "Node" interface. The elements may also contain other nested sub-elements that are descendants of the parent elements.

<Element_example id="example">

<subelement>text</sub-element>

</Element_example>

The "Attr" interface defines the attribute that is associated with the element. The attributes form separate nodes and are not descendants of the element. Within the example shown herein, the attribute is expressed by string id="example" [15,16].

The "Document" interface represents initial node, but cannot be considered as the root element. However, the most important interface is "Text", which carries values of the elements themselves, and is the only descendant of the element interface [16].

2.3. Document Segmentation Using the DOM

Most websites are built on a structured layout, where the content of the page is organized into defined segments. Some segments, such as navigation bars, headers, or footers, increase a page's usability and sell it with a consistent look.

On the other hand, elements such as advertising blocks distract the visitor but also serve as a source of revenue for the operator. Thanks to the use of markup languages and the Document Object Model, a web or XML document can be interpreted in the form of a tree structure [17,18].

DOM Tree Structure

The DOM tree structure, also called the tag tree, is a hierarchical representation of the nested tag structure of HTML pages or XML documents [17].

As an example, consider the simple source code of a webpage in HTML, which contains a first-level heading and a table filled with bold text in a single column (see Figure 2):

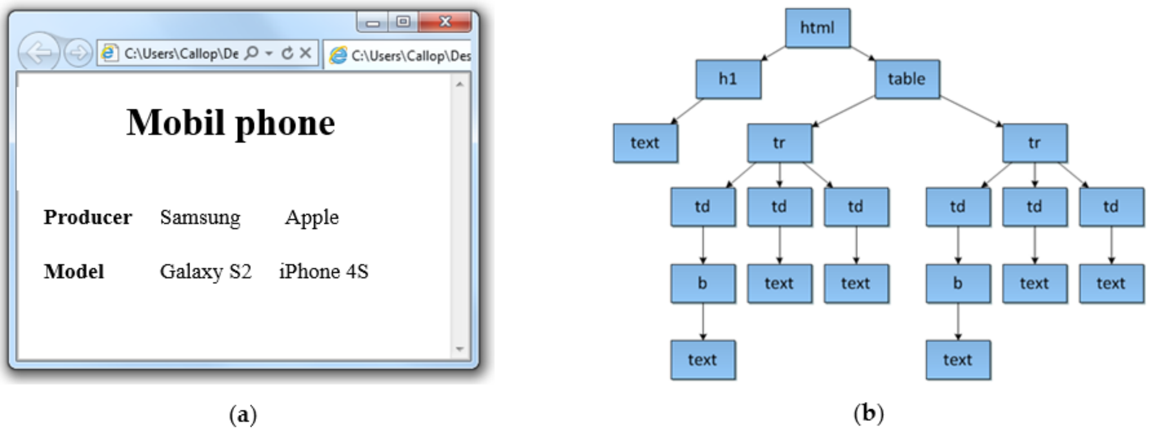


Figure 2. The example of representation of the source code by: (**a**) a web browser; (**b**) DOM tree structure.

```
<html>
<h1>Mobil phone</h1>
<table>
     <tr>
          <td><b>Producer</b></td>
          <td>Samsung</td>
          <td>Apple</td>
     </tr>
     <tr>
          <td><b>Model</b></td>
          <td>Galaxy S2</td>
          <td>iPhone 4S</td>
     </tr>
</table>
</html>
```

The web browser will display the code as shown in Figure 2a. A closer look at the source code reveals hierarchical structure. This is represented by the DOM tree, as shown in the model in Figure 2b, where each tag corresponds to a specific node in the model.

3. Natural Language Processing

Natural language processing (NLP) is an area of research that deals with the utilization of computers to understand and manipulate text or speech in natural language for further use. The aim is to acquire knowledge on the use of language to such an extent that tools and techniques for computer systems for natural language processing can be developed to perform desired tasks [19].

3.1. Understanding Natural Language

The core of all tasks in NLP consists of understanding natural language. The process of creating algorithms for understanding natural language includes three main problems:

- Understanding of thought processes;
- Representation and meaning of language input;
- Knowledge of words.

NLP system can therefore start from the level of words to determine the morphological structure and meaning of words. It may then progress to the level of sentences to determine the order of words, grammar, and meaning of the whole sentence. The final phase involves finding the context of the whole area. A given word or sentence may have a specific meaning if it is in the vicinity of other specific sentences and may depend on them semantically [19].

To understand natural languages and determine meaning from text or speech, it is important to distinguish between the seven interdependent levels that people use [19]:

3.1.1. Phonology and Morphology

This level deals with the interpretation of phonemes within and through words. Three types of rules are used in the phonological analysis:

- Phonetic rules—for sounds within words;
- Phonemic rules—for variations in pronunciation when words are spoken together;
- Prosodic rules—for fluctuations in strength and intonation throughout the sentence.

In language processing systems, speech input in the form of sound waves is converted to a digital signal and further analyzed. This level is not used for the text processing [20]. Within the morphological level, the component nature of words is examined. Words are composed of morphemes—the smallest parts of words that have material meaning. Take the word "unbreakable" as an example. It consists of three morphemes: the prefix "un", the root "break" and the suffix "able". The NLP system can use such a division of words into morphemes to determine the nature of the word. If the word 'able' is found in the word, it determines that it is an adjective relating to another word in the sentence [20].

3.1.2. Lexicon

The NLP system, similarly to humans, interprets the meaning of individual words at this level. At the same time, words that have only one possible meaning can be replaced by a semantic representation of that meaning. The nature of this representation depends on the semantic theory used in the NLP system [20].

The lexicon level may also require a dictionary. The NLP system can then determine when the dictionary will be used, as well as what scope and type of containing information will be used to describe the meaning of the word [20].

3.1.3. Syntax and Semantics

This level focuses on the analysis of words in a sentence and reveals the grammatical structure of the sentence. The output of the parsing is the representation of a sentence in the form of a structure of the dependence of relations between individual words. Syntax expresses meaning in most languages, because it is the order and relationships between words that contribute to the meaning of a sentence.

The semantic processing investigates the possible meanings of a sentence with a focus on the analysis of the relationships meanings between individual words in the sentence. This level of processing also involves the semantic removal of multiple meanings of a word and its definitive determination. If the rest of the sentence is also needed to determine the word's meaning, the syntactic level is not used, but the semantic level is taken into consideration [20].

3.1.4. Interview and Purpose

While the syntactic or semantic level is sufficient to understand natural language within a sentence, the NLP conversation level works with units more extensive than one sentence. The level focuses on the properties of the text as a whole and thus expresses the importance of using the connection between sentences. Furthermore, the conversation level recognizes properties of the text structure (introduction, core, conclusion, cited literature) and accordingly assigns a weight of meaning to certain parts [19,20].

This level concerns the effectiveness of language usage in certain situations and utilizes context beyond the content of the text to explain the meaning [20].

4. Design of an Algorithm for Main Text and Images Extraction from Web Documents

Since the mid-1990s, several surveys have shown the growing popularity of network services in common communication and finding information. Especially electronic mail and web have noticed increase in the search for working but also non-working content [21].

Therefore, there is a need to create algorithms for automatically processing data from web documents and retrieving valuable data from them. However, there are several problems with this task. Because the web is a free distribution system, the authors of content have been given an almost entirely free hand in the structure of web pages creation. The syntax of the main markup and programming language in HTML web documents can be broken or combined at the discretion of many programmers [22].

The automatic processing system must therefore correctly identify the main content of the web document and obtain data from it in the form of text and associated images. There are several systems for accessing and processing web documents. In this article, we propose an algorithm that is based on a combination of the DOM and NLP.

4.1. Combination of DOM and NLP Analysis

The technique for text and images extraction from a web document using both the DOM analysis and NLP was described by Parag Mulendra Joshi and Sam Liu of Hewlett-Packard Laboratories in Web Document Text and Images Extraction using DOM Analysis and Natural Language Processing [23].

The algorithm in the first phase extracts the body of the article from the input file (the web document). Subsequently, it uses semantic similarity based on NLP to find relevant images, which belong to the article.

4.1.1. Text Extraction

Each input file, that represents a web page, is mapped to the DOM for efficient analysis and further page content manipulation. As was already mentioned, the DOM has a hierarchical structure, with each tag in the document corresponding to one node in the tree interpretation of the model.

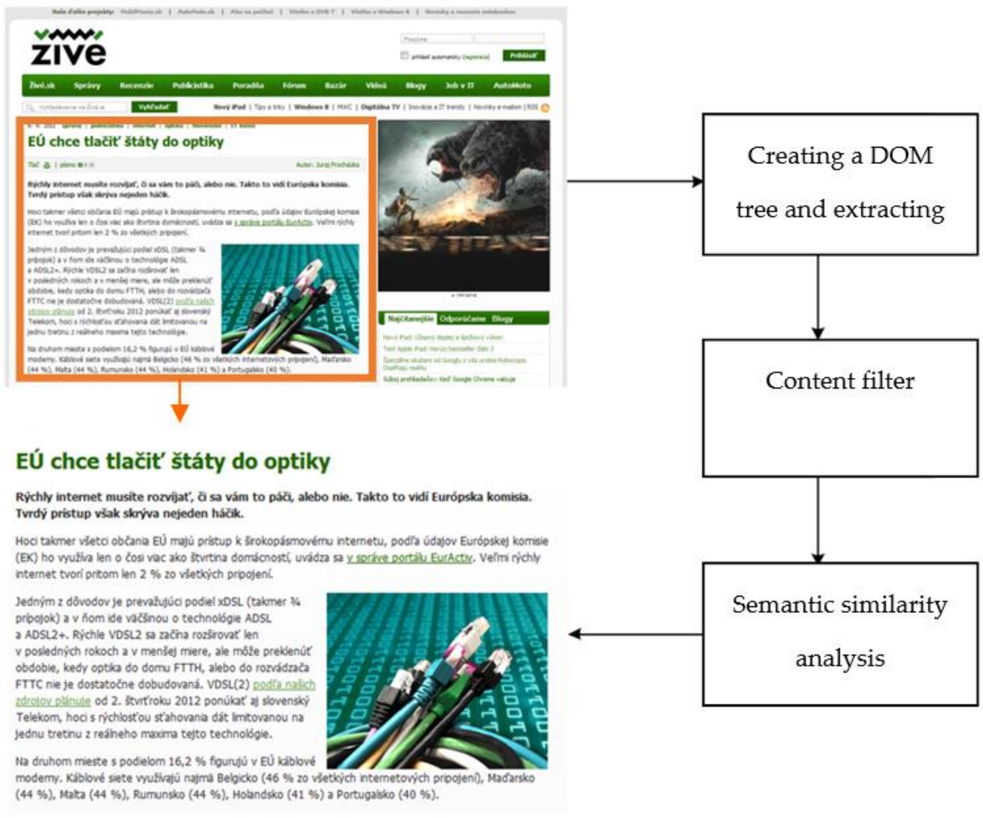
The body of a typical web article is coherent and divided into paragraphs as shown in Figure 3. However, the apparent visual structure does not always correspond to the DOM structure. This occurs because individual paragraphs can belong to different levels of the DOM subtree. Therefore, the aim of the article body extraction is to find out how the individual paragraphs form a whole block of text [23].

In practice, we encounter three ways to divide the text into paragraphs in HTML documents [23]:

- <div> tag
- <p> tag
- <br> tag

The <div> and <p> tags wrap the text in blocks, which are then represented by the browser as a separate paragraph. However, these may not be paired, which means that a <p> tag is used at the beginning of the block and an ending </p> tag at the end. The <br> tag is used within these blocks as a forced line break [23,24].

A typical article may contain one or more of such blocks that fall within the main block of text. In this proposal, the number of characters rule is used to find out which blocks can be identified as a part of an article. Firstly, all the subtrees that contain much text in the paragraphs are found. Subsequently, only those with more than 500 characters are considered, which will represent threshold value. From the processing point of view, the article is further proceeded only if consists at least 500 characters of text [23]. If the condition of a large number of characters is met by several subtrees, the first one occurring in the DOM structure is selected. Assuming that the main block of text is in the highest position in the web document other larger text blocks will follow it [23].


Creating a DOM tree and extracting
Content filter
Semantic similarity analysis

EÚ chce tlačiť štáty do optiky

Rýchly internet musíte rozvíjať, či sa vám to páči, alebo nie. Takto to vidí Európska komisia. Tvrdý prístup však skrýva nejeden háčik.

Hoci takmer všetci občania EÚ majú prístup k širokopásmovému internetu, podľa údajov Európskej komisie (EK) ho využíva len o čosi viac ako štvrtina domácností, uvádza sa v správe portálu EurActiv. Veľmi rýchly internet tvorí pritom len 2 % zo všetkých pripojení.

Jedným z dôvodov je prevažujúci podiel xDSL (takmer ¾ prípojok) a v ňom ide väčšinou o technológie ADSL a ADSL2+. Rýchle VDSL2 sa začína rozširovať len v posledných rokoch a v menšej miere, ale môže preklenúť obdobie, kedy optika do domu FTTH, alebo do rozvádzača FTTC nie je dostatočne dobudovaná. VDSL(2) podľa našich zdrojov plánuje od 2. štvrťroku 2012 ponúkať aj slovenský Telekom, hoci s rýchlosťou sťahovania dát limitovanou na jednu tretinu z reálneho maxima tejto technológie.

Na druhom mieste s podielom 16,2 % figurujú v EÚ káblové modemy. Káblové siete využívajú najmä Belgicko (46 % zo všetkých internetových pripojení), Maďarsko (44 %), Malta (44 %), Rumunsko (44 %), Holandsko (41 %) a Portugalsko (40 %).

Figure 3. The example of an article extraction from a web document.

4.1.2. Image Extraction

To use the semantic similarity test, it is essential to obtain individual images from a web document. Assuming that the text-related images are in the same subtree as the article's main text, a search of this subtree for the occurrence of <img> tags, which are used in HTML documents to mark embedded images, is performed. All text contained in the investigated subtree will be used to describe the images appearing in it.

From this text, the names of entities, such as the names of people or the names of places or organizations, are then recognized using a similarity algorithm. The algorithm then compares the obtained patterns of named entities with any two blocks of text to determine the similarity of the content.

A shortcut is created for small blocks with a small occurrence of named entities, which most often occurs in a larger block of content. For large blocks, the algorithm calculates the occurrence of each entity in both blocks of content. Subsequently, the density of the distribution of entities is normalized, and the maps standardize their occurrence in multidimensional space with the number of entities representing the dimension number.

From this, cosine similarity is derived based on two normalization frequencies of the distribution of entities from two text blocks. The angle between two frequency-distributed vectors represents it. Orthogonal vectors have a cosine degree of similarity of 0, which means zero similarity. On the contrary, the perfect similarity is indicated by vectors in the same direction. It is represented by the cosine degree of similarity 1 [23].

5. Algorithm for Main Content Extraction from Web Documents

The main task of the proposed algorithm is to identify and extract the main block of a web document that contains the text of the article and the relevant images. This chapter will describe the procedure of the algorithm, which uses the DOM for navigation in the document and, at the same time, specific characteristics of the natural language for the identification of the main block. Using the algorithm, the functionality of the selected procedure will be practically verified on a sample of 45 web documents of various types.

5.1. Identification of the Main Block of the Web Document

There are several ways to identify the main block of a web document with the desired content using the DOM. The main documents are:

- Known element identifier name (e.g., <div id="main_block">);
- Character counter;
- Sentence counter.

The simplest and most accurate method is to fix the block containing the main text and images. As mentioned in previous chapters, the <div> element carries a unique identifier in addition to its content. If the name of the attribute of an element that is represented in the web browser as the main block and carries the desired content is known, this content can be easily extracted using the proposed algorithm.

However, this method is unsuitable for the automatic extraction of the main block of various documents. This is because HTML documents do not have fixed identifier names that carry the main content. The method is used mainly on pages with content focused on news. Therefore, the algorithm must be able to find the main block of text automatically.

Another way is to use a character counter to identify the main text. The main article is the one that consists of at least 500 text characters. If there is more than one such block in the document, only the first one is considered.

In practice, however, this simple character counter did not work for us. Many web documents contained several blocks with more than 500 characters, with the main text only in the first place. Dynamic websites also use long text strings to create drop-down menus or similar interactive blocks, which at a time drew only a part of the string using JavaScript as moving text or a list. Therefore, utilization of a sentence counter in the proposed algorithm was chosen.

Image Extraction

Addressing above mentioned, it is more appropriate to use a certain characteristic of natural language to distinguish the relevant text from the auxiliary text strings. Instead of a simple counter of characters or even words, which has the same disadvantages, the counter of whole sentences is chosen. The characteristic of sentence structure from natural language is taken; each new sentence follows the final punctuation mark of the previous sentence (dot, question mark, exclamation mark) and a space.

It was observed that most common articles do not use a question mark or exclamation mark to end sentences to such an extent as to affect the identification of the main article. Therefore, the string "dot + space" as the separating rule of the new sentence is used.

After loading the document, the text content of each element for the number of sentences by the given rule is tested. The algorithm will consider the relevant text as a block containing at least five sentences. Contrary to the previous algorithm, in this case, we will not end up testing the elements on the first block found. This is because some web documents divide the main article block into several elements, between which other unwanted content can be placed. The resulting extracted text is thus composed of all sub-blocks that meet the condition in question.

Elements that hide suitable text with relevant content are also tested for the presence of images. We assume that the images related to the text are in these blocks. If their presence is confirmed, the algorithm ensures their extraction simultaneously with the text.

The procedure for identifying the main text in the form of a flowchart is shown in Figure 4.

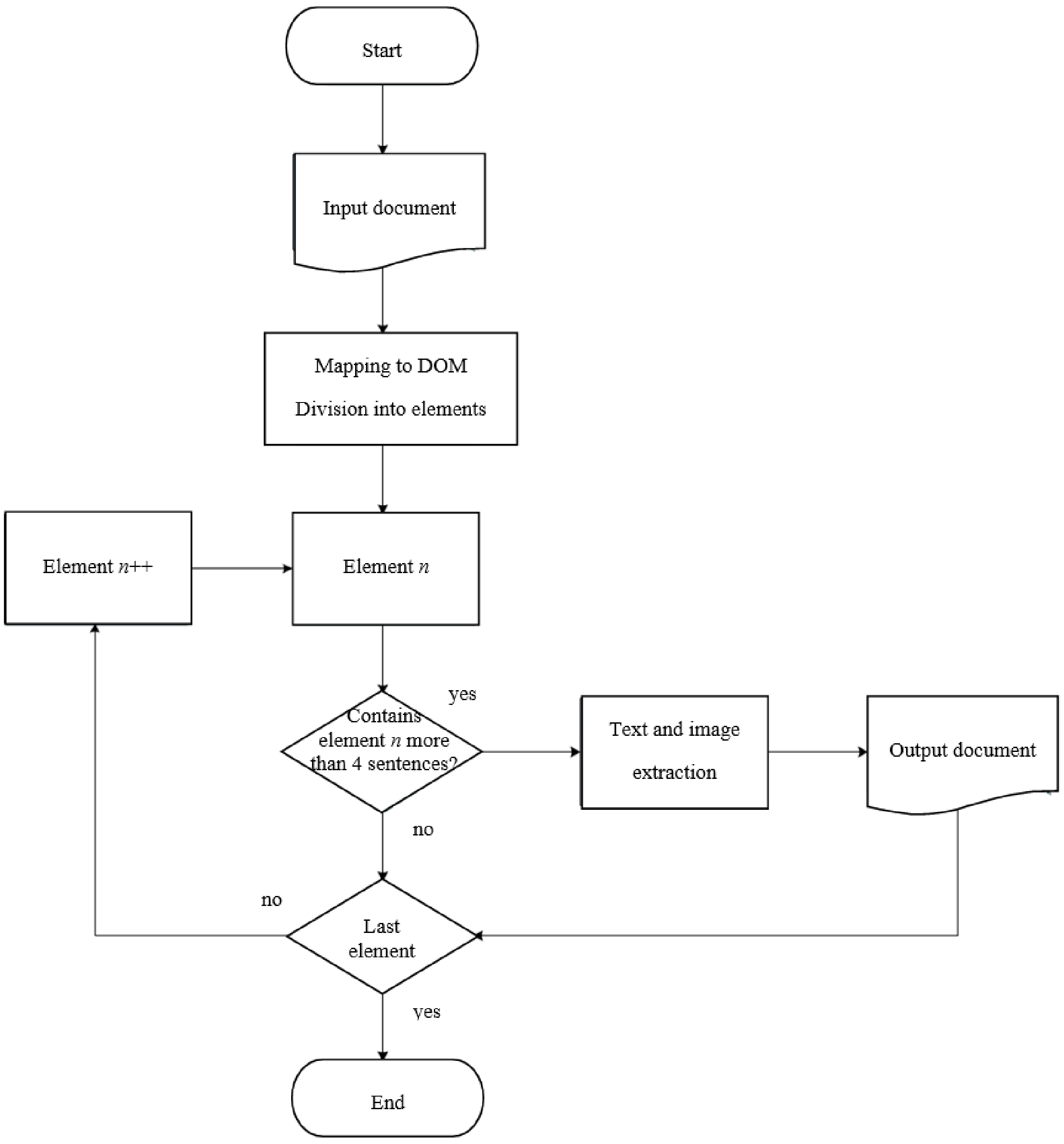


Figure 4. The flowchart for identifying the main block of a web document.

5.2. The Algorithm Proposal

To verify the functionality of the proposed procedure, an algorithm was developed and tested in a real environment we verified. The algorithm is written in the JAVA programming language within the Eclipse development environment. The programming language and environment were chosen with respect to further planned implementation, namely the conversion of the algorithm to a mobile application supporting android 10. During the algorithm creation, the following points were adhered to:

- Loading a document;
- Mapping a document to the DOM;
- Division into elements;
- Scan elements and test them with a specified filter;
- Text and image extraction;

- Save extracted material to external files to verify extraction accuracy.

5.2.1. Loading a Document

JSOUP library is chosen for loading and mapping the input document to the DOM and working with it. The library is published with open source and provides an application interface for retrieving an HTML document from a URL, a local file, or a text string. It implements the WHATWG HTML5 specification, thus providing an identical DOM model as modern Internet browsers.

It also allows easy navigation in the document using the names of the identifier, brand, class, or attribute of individual elements, which would allow utilization of methods for specific data extraction from them. The library supports methods that return attribute names, identifiers or class names, plain text or formatted HTML code content, or additional data in the form of scripts or cascading styles from the selected element.

When loading a web document in HTML format, two ways for loading it were considered:

- Loading using a URL from external storage;
- Loading from a file on local storage.

The JSOUP library implements an easy way to retrieve a document from a URL using the following command:

```
Document doc = Jsoup.connect(“http://url_adresa.ext/stranka.html”).get().
```

To load a document from local storage, one must also use the java.io.File library, which contains the methods needed to work with files. Subsequently, the connect () and get () methods from the JSOUP library are replaced with the parse () method. The resulting code for loading the document is as follows:

```
File input = new File(“absolute path to the file.html”);

Document doc = Jsoup.parse(input,null).
```

When working with files, it is important to pay attention to the correct encoding. The JSOUP library automatically assigns encoding according to meta tags when retrieving documents from a website. However, when using local files, you must manually assign the correct encoding to the parse () method or set it to “null”, as shown in the example. In this case, the algorithm will behave as when loading a document from a website, and thus, the correct encoding from the document is determined automatically.

After successful document loading, the library will ensure its mapping to the DOM model and access to it using the appropriate methods.

5.2.2. Search and Test Elements

The next step to identify the main block of a web document is to search all elements for the presence of text according to the selected condition. In general, web pages can be formatted directly using HTML code or cascading styles.

The second mentioned variant requires “wrapping” the respective blocks of a certain format into neutral <div> tags. Most modern web documents use this type of formatting. However, with the intention to cover older documents as well, the proposed algorithm had to be adapted to search for blocks in both types of documents.

The navigation in the document is solved by a two-phase search for elements by brand name, as follows:

- Searching in blocks <div>;
- Extension of the block searching <body>.

Thus, in the initial phase, the algorithm first tries to find the text according to the selected condition in blocks bounded by neutral <div> tags. If no matching block is found, the algorithm extends the search to the entire body of the document, which is bounded by a

<body> tag. This covered not only modern dynamic websites, which are mostly generated by a publishing system using cascading styles, but also older or static documents with simple formatting.

The condition that verifies whether or not the relevant text is in the element has already been mentioned. Therefore, the relevant content matching is considered only if it contains text with a minimum length of five sentences. The following code handles text division and testing:

```
String[] rtext = content.text().split("\\. ");
if(rtext.length >4){
//export
}
```

The "content" variable carries the current content of the selected element. In the field of text strings "rtext", individual sentences divided in accordance with the parameter "." (dot + space) are then assigned.

If the field reaches a size of more than four strings, thus fulfilling the condition with a minimum content size of five sentences, the algorithm executes additional commands to export text and images to external files. This process is repeated until the algorithm encounters the last element in the document to search for other suitable blocks of text.

5.2.3. Duplication Treatment

When extracting a single document multiple times, a situation where the algorithm seemingly extracts one text block several times could occur. Such a case will result in duplications in the output file. This is because one text block can be nested inside several other elements. For example, higher-level elements can wrap this text in title blocks, additional information blocks, and the like.

To address the problem with duplication, double duplicate protection is implemented into the algorithm. In the first step, the algorithm verifies if the element starts with the text string "<div>", meaning that such an element is not finite and contains another child.

However, by testing in a real environment, it was found that this protection is only sufficient for some sites. Thus, the second protection step, which verifies the already directly extracted text, is implemented as well.

Suppose the algorithm finds that the content of another element in the document meets the set condition of the relevant text. In that case, it first verifies its content with a previously downloaded text block. If it finds a match, the element is skipped. Otherwise, it is appended to the external file after the last downloaded text block.

5.2.4. Export to External Files

The actual extraction of text blocks is simple, thanks to the selected JSOUP library. After successful navigation to a suitable element, which has passed the selected filter, it is sufficient to define the form of extracted content (whether we want to extract content from it in the form of plain text or formatted HTML code). For this purpose, JSOUP has implemented two methods (content.text () and content.html ()), where the variable "content" expresses the relevant element.

The algorithm exports extracted content to both types of files simultaneously. While using the text format, it is enough to save the selected text in a new file. With the HTML code, the external file has to be prepared appropriately. The first time you access it, the standard code for interpretation in a web browser is written at the beginning of the HTML file:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
<title>title</title>
</head>
<body>
```

As part of the first and each subsequent access to the file, the extracted HTML code from the input document is added to the code prepared in the described way. If the algorithm detects the last element, it appends the closing </body> and </html> tags to the file, which finalizes the external file preparation for use in web browsers.

During creation, both the output text and HTML files are assigned a name formed by the title of the input document, which means the algorithm retrieves the title from the <title> meta tags. The same procedure is valid also to extract images.

As already mentioned, the blocks that have been identified as the main blocks of the document are scanned for the presence of images simultaneously with text extraction. The algorithm scans *jpg, *png, *gif, *bmp, and *tiff files from the HTML code. Once the file is found in the investigated block, a folder named as the title of the input document is created, followed by the string "_files". All found images are then saved in it.

However, the caption must be filtered for illegal characters. If file or folder names contain such characters, the operating system will not allow them to be created. Illegal characters are a colon, question mark, slash, asterisk, or angle brackets. In the resulting title, which is used as file and folder names, these characters are replaced by a space.

6. Testing of the Algorithm

The proposed algorithm is tested in a real environment on a sample of 45 web documents. Testing consisted of the following steps:

- Creating a database of web documents;
- Extraction of the main block using an algorithm;
- Analysis of the results.

6.1. Creating a Database of Web Pages

Thorough testing is necessary for the objectification of algorithm evaluation. For this purpose, a website database that contains several types of documents has been created, consisting of both simple static documents and complex pages generated by publishing systems, as well. The documents are divided into the following categories according to the content topics: water beavers, ticks, fungi, dinosaurs, flamingos, dogs, elephants, ants, scorpions, and butterflies. All tests were performed on pages in the Slovak language. Each of the topics is represented by an average of five websites. The pages are made up of text and images that thematically agree with the content. All documents then went through the created algorithm.

6.2. The Evaluation of the Algorithm

After the proposed algorithm is applied to chosen pages, the output files are compared with the desired state and, thus, the real content of the main block of the web document. The compliance (matching) of the text output of the algorithm with the manually marked main text of the original document is then evaluated.

For comparison, a character counter programmed in the JAVA programming language is developed together with the algorithm for document processing. The counter first removes all spaces from the input document to handle the different text formatting. Subsequently, the application counts all the characters of the text. The results of the process are recorded in Table 1.

From the resulting data, the percentage difference between char counts is calculated in the third column of Table 1. For better representation, the values are also shown graphically in Figure 5.

Table 1. The list of tested websites and extraction results (in Slovak language).

Number	HTML	Number Char		Difference
		Required	Acquired	
1	bobobobor.html	4414	4414	0.00
2	14bobor.html	10,360	12,008	15.91
3	bobor-vodny.htm	7139	7165	0.36
4	ZVIERATA—bobor vodny.htm	1871	2418	29.24
5	Bobor vodný—Časopis Poľovníctvo a rybárstvo.htm	1752	2423	38.30
6	Kliešť obyčajný (Ixodes ricinus).html	9345	9896	5.90
7	Kliešť Takto sa zahryzne Topky sk.html	2615	2678	2.41
8	Cassovia.sk Kliešte aj v zime. Rozhovory a reportáže.htm	3982	4857	21.97
9	Kliešte v meste Správy Správy Rodinka.sk.htm	4447	4547	2.25
10	Kliešte ohrozujú najviac seniorov, chýba im očkovanie—Zdravie a prevencia—zdravie.pravda.sk.htm	2194	2401	9.43
11	Jedovaté huby.html	4611	5176	12.25
12	Huby.htm	1251	1251	0.00
13	Huba mesiaca júl 09—Časopis Poľovníctvo a rybárstvo.htm	1485	2160	45.45
14	Do lesa s dozimetrom Máme sa báť zbierať huby—Život.sk.htm	5092	5219	2.49
15	Pozor na huby na tanieri.htm	2825	2951	4.46
16	Najnovší objav vedcov Dinosaurus s vráskavými očami—ADAM.sk.htm	1713	1443	−15.76
17	www.kutilas.estranky.sk—Dinosaury.htm	7585	7981	5.22
18	infovekacik.htm	3044	3195	4.96
19	eQuark.sk—portál pre popularizáciu vedy—Prvý dinosaurus s jedným prstom.htm	1867	1882	0.80
20	Dinosaurus bol iný, ako sme si pôvodne mysleli Veda a technika-články 16-06-2011 Veda a technika Noviny.sk.htm	1424	1505	5.69
21	Plameniak ružový ZOO Bojnice.htm	1000	1329	32.90
22	plameniaky.htm	1451	1914	31.91
23	Plameniaky na ružovo farbia baktérie a beta karotén Zaujímavosti prievidza.sme.sk.htm	1882	1856	−1.38
24	Hadogenes paucidens.html	2535	2441	−3.71
25	eQuark.sk—portál pre popularizáciu vedy—Jed škorpiónov je vhodný do pesticídov.htm	1261	1276	1.19
26	História psa Pes-portál.sk.htm	2195	2195	0.00
27	Poľovníka postrelil pes.htm	980	778	−20.61
28	stvornohykamarat—Plemená psov na C a Č—Český strakatý pes.htm	1732	2538	46.54
29	Aj pes, ktorý šteká, hryzie. Nebezpečne—Zdravie a prevencia—zdravie.pravda.sk.htm	6596	6771	2.65
30	DOBERMAN—História dobermana—DOBERMAN3.htm	5406	6198	14.65
31	Slon africký ZOO Bojnice.htm	2786	3307	18.70
32	Je zaujímavé aká môže byť príroda že—Fotoalbum—Cicavce—Slon africký.htm	1793	2240	24.93
33	Ivan Pleško—O slonoch—Slon Africký (Loxodonta Africana).htm	13,539	13,541	0.01
34	Zvieratká—Suchozemské zvieratá—Slon Africký.htm	4966	5336	7.45
35	MARŤANKOVIA.htm	1371	1475	7.59
36	Vyspelá civilizácia mravcov.htm	14,181	15,815	11.52
37	rad Blankokrídlovce—Mravce.htm	3203	3228	0.78
38	Mravce používajú antibiotiká. Pestujú huby a spolupracujú s baktériami Biológia veda.sme.sk.htm	3837	3731	−2.76
39	Mravce—etológia, biológia a ich chov—článok zo serveru www.vivarista.sk.htm	12,426	12,578	1.22
40	Mravce. Blog—Michal Wiezik (blog.sme.sk).htm	6793	8141	19.84
41	Atlas živočíchov vidlochvost ovocný—Na túru s NATUROU.htm	2060	2126	3.20
42	Moľa DDD služby.htm	1478	1491	0.88
43	Vidlochvost Feniklový Motýle Slovenskej republiky.htm	1763	2017	14.41
44	Babôčka osiková « CASD Liptovský Mikuláš.htm	1533	1937	26.35
45	Hnedáčik Pyštekový Motýle Slovenskej republiky.htmr.html	1838	2092	13.82

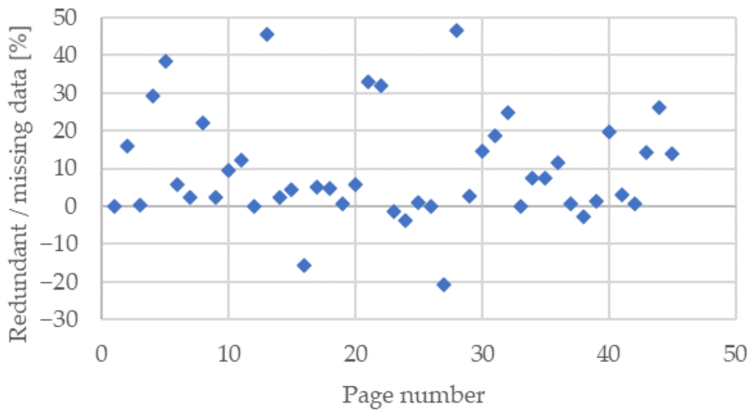


Figure 5. The graphical display of the results of the algorithm (comparison of the desired state with the output of the algorithm).

From the results is clear that more than half of the tested documents, represented by standard articles with a range of 5 to 10 paragraphs, showed a difference of less than 10% of the extracted content from the counted one. This means a negligible deviation, which is related either to a block of social networks or additional content located in the main block.

To express the efficiency of the algorithm Ea, the following relation is used:

$$Ea = 100 - \frac{\sum_{i=1}^{Nc} \left| \left(\frac{Nr_i}{Na_i} * 100 \right) - 100 \right|}{Nc} \, [\%] \, , \tag{1}$$

where Nc expresses the total number of documents, Nr_i is the number of real characters of the main document block, and Na_i is the number of characters of the output document from the algorithm. After substituting the data into relation 1, the overall efficiency of the algorithm would be:

$$Ea = 100 - \frac{531.8843}{44} = 88.18 \, [\%] \tag{2}$$

The proposed algorithm resulted in an overall efficiency of 88.18% for text extraction on a selected sample of documents. Image extraction depends on the identification of the main block of the document, so it successfully copies the state of text extraction.

However, the effectiveness depends on the selected sample of documents. As can be seen from the graphical representation of the results, the variance of the differences between the useful and extracted content is relatively large. While some types of pages show complete matching with extracted content, other types provide up to half as much useless content. Thus, the efficiency may vary significantly with a different sample of documents. The big differences are due to the unification of the website since the structure of the source code can differ significantly.

6.3. Comparison of Experimental Results

In the last five years, research in extracting information from web pages has rapidly developed. From Table 2, it can be seen that there are many researchers who are dedicated to the issue and achieve very suitable results with comparable performances. However, each of the authors focuses on a different extraction of information from the document, and therefore the results could be considered completely different, even if, in some cases, they achieve the same success rate. The differences could be found not only within the extraction of information but also within the practical implementation.

Table 2. The comparison of the web page classification approaches.

Author	Year	Approach	Technique	Classification Accuracy [%]
Min Gu, Feng Zhu, Qing Guo, Yanhui Gu, Junsheng Zhou, Weiguang Qu [25]	2016	Towards Effective Web Page Classification	SVM	84.15
Xin Yu, Zhengping Jin [26]	2017	Web Content Information Extraction Based on DOM Tree and Statistical Information	DOM, CEDS	97.90
Nichita Utiu, Vlad-Sebastian Ionescu [27]	2018	Learning Web Content Extraction with DOM Features	LR, SVM, DT, RF, MLP	80.85
Gurjyot Singh Kalra, Ramandeep Singh Kathuria, Amit Kumar [28]	2019	YouTube video classification based on title and description text	Random Forest : Decision tree	98.00
A. Poornima, k. Sathiya Priya [29]	2020	A comparative sentiment analysis of sentence embedding using machine learning techniques	Support Vector Machine, Random Forest, Naïve Bayes	86.00
Martina Radilova, Patrik Kamencay, Robert Hudec, Miroslav Benco, Roman Radil	2022	Tool for Parsing Important Data from Web Pages	DOM, NLP	88.18

In the article [25], the authors propose a novel web page classification strategy integrating topic model and SVM, with the intention to manage and organize information on the web. They use the topic model to harness the implicit information on web pages for feature extraction. With the latent information obtained from multi-LDA, the relation between features and documents becomes closer. The accuracy of the strategy is 84.15%.

Another research group, in work [26], propose a method that can adapt to the heterogeneity and variability of web pages, providing high precision and recall. Their method is based on the DOM structure to divide one web page into several blocks and extract content blocks with statistical information instead of machine learning repeating training and manual labeling, which resulted in high performance in terms of precision, recall, and F1.

The paper [27] presented a method of web content extraction inspired by current popular methods, using machine learning together with features based on information from the DOM tree. Through experimental evaluation, authors have shown competitive results on the Cleaneval dataset with results achieved by our proposed algorithm that we achieve. On the Dragnet dataset, the proposal described herein outperforms other state-of-the-art methods with all their models, even with a computationally cheap logistic regression. This result persists even during testing on the entire set of tags of a page as opposed to a reduced set of candidates as other papers do. When testing on the entire dataset for Dragnet, one can observe that differences between model performances are exacerbated. They cannot tell whether this is due to the limitations of the models or because of the experimental design, as their use of a less statistically sound model selection can be more prone to model instability.

Other groups in their works [28,29] focused on video record extraction. YouTube has a huge amount of video, and maintaining the records of the video for use and efficient retrieval is very complex and difficult. The proposed approach is to extract the video record by selenium and Beautiful Soup libraries and divide the data collection into test and train data collection. The classification of the data collection of videos is performed by random forest classifier along with natural language processing techniques based on title and description. After classification using a different computational matrix, the performance of the given model achieves 98% accuracy.

There is a large number of reviews, comments, and feedback on social media about people's experiences with the event or product, and people check others' experiences, whether it is positive or negative about the same. The proposed approach defines classification based on sentiment analysis, which means classifying the comments by feature label as positive and negative. This sentiment analysis contains all format data such as structured, unstructured, and semi-structured. The sentimental analysis was carried out by examining various classification algorithms' performance. In the given paper, multinomial naive Bayes, logistic regression, and support vector machine (SVM) algorithms classification performance compared among them logistic regression achieve 86% accuracy [29].

7. Discussion

In this paper, the data from web documents using various techniques were extracted. The extracted data needs to be classified so that we obtain the relevant data. This article describes our own algorithm based on DOM and NLP techniques but also other approaches for extracting information from web pages using various classification techniques such as support vector machine, decision tree techniques, naive Bayes, and K-nearest neighbor. The classification of individual methods is drawn based on articles from individual authors who carried out the research. Neural networks and decision tree techniques achieve better results in accuracy compared to other methods, even when the web data contains noise.

The best results, according to the articles, were achieved by Xin Yu, Zhengping Jin and Gurjyot Singh Kalra, Ramandeep Singh Kathuria, and Amit Kumar over 90%. Excellent results are also achieved by what specifically he decides to extract. In the case of the author's collective in the article YouTube video classification based on title and description text, they extract the video based on the title and description text that the author of the website provides in the code. In their conclusions, the authors assess that the results are biased on the test set of web pages, as nowadays, the authors of web pages have a free hand in writing the code.

8. Conclusions

In this paper, the algorithm for extracting the main block (textual and image data) of the web document was proposed. As part of our future work, we plan to expand the application published at the TSP conference [13] with the proposed algorithm. In this way, the user could automatically parse the web page after clicking the parse button without manually marking the main block, as current applications do.

The preparation of the work is also a theoretical view of the object-oriented representation of the web document and natural language processing. In the article, the current state of web document processing was presented. While the algorithms that reorganized the document were made more suitable for the semantic description of images, the method with a combination of the Document Object Model and natural language processing was suitable for our main block extraction purposes. A similar principle in the proposed procedure for extracting a document was used. The classification accuracy of the proposed algorithm programmed in JAVA was approximately 88%. The sample of 45 documents contained both static and dynamic document types.

The main contributions of the work are in the design of the algorithm for the identification of textual and image information from web documents. The proposed algorithm is suitable for further processing of output documents. It can be used in the semantic description of images but also as a basis for computer and mobile applications in the field of web document processing. In our future work, the resulting performance of the proposed method will be improved using machine learning algorithms and the application of deep neural networks. We plan to replace the technique for extracting textual and image data from web documents with a deep neural network.

Author Contributions: Conceptualization, M.R., P.K. and R.H.; methodology, M.R. and P.K.; software, M.R.; validation, P.K., M.B., R.H. and R.R.; formal analysis, M.R., P.K. and R.R.; investigation, M.R.; data curation, M.R.; visualization, M.R., P.K. and R.R.; supervision, M.R. and P.K.; project administration, R.H. and P.K.; funding acquisition, R.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Slovak Research and Development Agency under contract no. APVV-21-0502: BrainWatch: System for automatic detection of intracranial aneurysms and by project PP-COVID-20-0100: DOLORES.AI: The pandemic guard system.

Institutional Review Board Statement: Ethical review and approval were waived for this study.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are available from the authors upon request. This is according to laboratory rules.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ferrara, E.; De Meo, P.; Fiumara, G.; Baumgartner, R. Web data extraction, applications and techniques: A survey. *Knowl.-Based Syst.* **2014**, *70*, 301–323. [CrossRef]
2. Uzun, E. A Novel Web Scraping Approach Using the Additional Information Obtained from Web Pages. *IEEE Access* **2020**, *8*, 61726–61740. [CrossRef]
3. Najafabadi, M.M.; Villanustre, F.; Khoshgoftaar, T.M.; Seliya, N.; Wald, R.; Muharemagic, E. Deep learning applications and challenges in big data analytics. *J. Big Data* **2015**, *2*, 1. [CrossRef]
4. Figueiredo, L.N.L.; de Assis, G.T.; Ferreira, A.A. DERIN: A data extraction method based on rendering information and n-gram. *Inf. Process. Manag.* **2017**, *53*, 1120–1138. [CrossRef]
5. Kushmerick, N. *Wrapper Induction for Information Extraction*; University of Washington: Seattle, DC, USA, 1997.
6. Liu, L.; Pu, C.; Han, W. XWRAP: An XML-enabled wrapper construction system for Web information sources. In Proceedings of the 16th International Conference on Data Engineering (Cat. No.00CB37073), San Diego, CA, USA, 29 February–3 March 2000; IEEE Computer Society: Washington, DC, USA, 2000; pp. 611–621.
7. Das, R.; Turkoglu, I. Creating meaningful data from web logs for improving the impressiveness of a website by using path analysis method. *Expert Syst. Appl.* **2009**, *36*, 6635–6644. [CrossRef]
8. Fazzinga, B.; Flesca, S.; Tagarelli, A. Schema-based Web wrapping. *Knowl. Inf. Syst.* **2011**, *26*, 127–173. [CrossRef]
9. Kao, H.Y.; Lin, S.H.; Ho, J.M.; Chen, M.S. Mining web informative structures and contents based on entropy analysis. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 41–55. [CrossRef]
10. Zachariasova, M.; Hudec, R.; Benco, M.; Kamencay, P. Automatic extraction of non-textual information in Web document and their classification. In Proceedings of the 2012 35th International Conference on Telecommunications and Signal Processing (TSP), Prague, Czech Republic, 3–4 July 2012; pp. 753–757. [CrossRef]
11. Li, Z.; Ng, W.K.; Sun, A. Web data extraction based on structural similarity. *Knowl. Inf. Syst.* **2005**, *8*, 438–461. [CrossRef]
12. Maghdid, H.S. Web News Mining Using New Features: A Comparative Study. *IEEE Access* **2019**, *7*, 5626–5641. [CrossRef]
13. Radilova, M.; Kamencay, P.; Matuska, S.; Benco, M.; Hudec, R. Tool for Optimizing Webpages on a Mobile Phone. In Proceedings of the 2020 43rd International Conference on Telecommunications and Signal Processing (TSP), Milan, Italy, 7–9 July 2020; IEEE: Milan, Italy, 2020; pp. 554–558.
14. Wood, L. Programming the Web: The W3C DOM specification. *IEEE Internet. Comput.* **1999**, *3*, 48–54. [CrossRef]
15. World Wide Web Consortium. *Document Object Model (DOM) Level 1 Specification*; World Wide Web Consortium: Cambridge, MA, USA, 1998.
16. Nadee, W.; Prutsachainimmit, K. Towards data extraction of dynamic content from JavaScript Web applications. In Proceedings of the 2018 International Conference on Information Networking (ICOIN), Chiang Mai, Thailand, 10–12 January 2018; pp. 750–754. [CrossRef]
17. Vineel, G. Web page DOM node characterization and its application to page segmentation. In Proceedings of the IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA), Bangalore, India, 9–11 December 2009; pp. 1–6. [CrossRef]
18. Luo, J.; Shen, J.; Xie, C. Segmenting the web document with document object model. In Proceedings of the IEEE International Conference on Services Computing (SCC 2004), Shanghai, China, 15–18 September 2004; IEEE: Shanghai, China, 2004; pp. 449–452.
19. Chowdhury, G.G. Natural language processing. *Annu. Rev. Inf. Sci. Technol.* **2003**, *37*, 51–89. [CrossRef]
20. Liddy, E.D. *Natural Language Processing*; Syracuse University: New York, NY, USA, 2001.
21. Savolainen, R.; Kari, J. Placing the Internet in information source horizons. A study of information seeking by Internet users in the context of self-development. *Libr. Inf. Sci. Res.* **2004**, *26*, 415–433. [CrossRef]
22. Shengnan, Z.; Jiawei, W.; Kun, J. A Webpage Segmentation Method Based on Node Information Entropy of DOM Tree. *J. Phys. Conf. Ser.* **2020**, *1624*, 032023. [CrossRef]
23. Joshi, P.M.; Liu, S. Web Document Text and Images Extraction using DOM Analysis and Natural Language Processing. In Proceedings of the 2009 ACM Symposium on Document Engineering, Munich, Germany, 16–18 September 2009; pp. 1–4. [CrossRef]
24. Alimohammadi, D. Meta-tag: A means to control the process of Web indexing. *Online Inf. Rev.* **2003**, *27*, 238–242. [CrossRef]

25. Gu, M.; Zhu, F.; Guo, Q.; Gu, Y.; Zhou, J.; Qu, W. Towards effective web page classification. In Proceedings of the 2016 International Conference on Behavioral, Economic and Socio-Cultural Computing (BESC), Durham, NC, USA, 11–13 November 2016; pp. 1–2. [CrossRef]
26. Yu, X.; Jin, Z. Web content information extraction based on DOM tree and statistical information. In Proceedings of the 2017 IEEE 17th International Conference on Communication Technology (ICCT), Chengdu, China, 27–30 October 2017; pp. 1308–1311. [CrossRef]
27. Utiu, N.; Ionescu, V. Learning Web Content Extraction with DOM Features. In Proceedings of the 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 6–8 September2018; pp. 5–11. [CrossRef]
28. Kalra, G.S.; Kathuria, R.S.; Kumar, A. YouTube Video Classification based on Title and Description Text. In Proceedings of the 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 18–19 October 2019; pp. 74–79. [CrossRef]
29. Poornima, A.; Priya, K.S. A Comparative Sentiment Analysis of Sentence Embedding Using Machine Learning Techniques. In Proceedings of the 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 6–7 March 2020; pp. 493–496. [CrossRef]