electronics

MDPI

Article

Toward the Implementation of Text-Based Web Page Classification and Filtering Solution for Low-Resource Home Routers Using a Machine Learning Approach

Audronė Janavičiūtė, Agnius Liutkevičius * and Nerijus Morkevičius

Department of Computer Sciences, Kaunas University of Technology, 44249 Kaunas, Lithuania; audrone.janaviciute@ktu.lt (A.J.); nerijus.morkevicius@ktu.lt (N.M.)
* Correspondence: agnius.liutkevicius@ktu.lt

Abstract

Restricting and filtering harmful content on the Internet is a serious problem that is often addressed even at the state and legislative levels. Existing solutions for restricting and filtering online content are usually installed on end-user devices and are easily circumvented and difficult to adapt to larger groups of users with different filtering needs. To mitigate this problem, this study proposed a model of a web page classification and filtering solution suitable for use on home routers or other low-resource web page filtering devices. The proposed system combines the constantly updated web page category list approach with machine learning-based text classification methods. Unlike existing web page filtering solutions, such an approach does not require additional software on the client-side, is more difficult to circumvent for ordinary users and can be implemented using common low-resource routers intended for home and organizations usage. This study evaluated the feasibility of the proposed solution by creating the less resource-demanding implementations of machine learning-based web page classification methods adapted for low-resource home routers that could be used to classify and filter unwanted Internet pages in real-time based on the text of the page. The experimental evaluation of softmax regression, decision tree, random forest, and linear SVM (support vector machine) machine learning methods implemented in the C/C++ programming language was performed using a commercial home router Asus RT-AC85P with 256 MB RAM (random access memory) and MediaTek MT7621AT 880 MHz CPU (central processing unit). The implementation of the linear SVM classifier demonstrated the best accuracy of 0.9198 and required 1.86 s to process a web page. The random forest model was only slightly faster (1.56 s to process a web page), while its accuracy reached only 0.7879.

Keywords: web page blocking; web page filtering; text-based classification; machine learning; performance evaluation; home router

Academic Editors: Pei-Chun Lin and Tarik Taleb

Received: 8 July 2025
Revised: 16 August 2025
Accepted: 18 August 2025
Published: 18 August 2025

Citation: Janavičiūtė, A.; Liutkevičius, A.; Morkevičius, N. Toward the Implementation of Text-Based Web Page Classification and Filtering Solution for Low-Resource Home Routers Using a Machine Learning Approach. *Electronics* **2025**, *14*, 3280. https://doi.org/10.3390/electronics14163280

1. Introduction

1.1. Background

Statistical data reveal that in 2025, the number of Internet users reached 5.56 billion [1], while the global average screen time reached an alarming 6 h and 38 min [2]. It is obvious that the Internet is considered one of the modern inventions that has caused a technological revolution that improves the quality of people's lives, but attention is increasingly being drawn to its negative side, like addiction, various types of cybercrime, distribution of

harmful content, unwanted advertising, etc. While offering great opportunities to learn and develop various skills, the information found on the Internet can be not only useful, but harmful as well. Children and young people are especially vulnerable to harmful content on the Internet. According to UNICEF (United Nations Children's Fund), children are usually exposed to threats such as cyberbullying, hate speech, and violent content, including messages that incite self-harm and even suicide [3]. With the expansion of digital services and the availability of Internet connectivity, the need to prevent the dissemination of illegal and harmful content is increasing. Therefore, various laws, regulations, and international initiatives are applied for this purpose. For example, the United States of America's Children's Internet Protection Act (CIPA) requires schools and libraries to verify that they have an Internet safety policy that includes technology protection measures, blocking or filtering Internet access to pictures that are as follows: (a) obscene; (b) child pornography; or (c) harmful to minors (for computers that are accessed by minors) [4]. The policies of the European Union (EU) are even more strict, requiring one to take measures to ensure a high level of privacy, safety, and security of minors. The children's online protection policies of the EU, summarized in [5], are related to a wide range of threats, including cyberbullying, disinformation and misinformation, promotion of unhealthy habits, fraudulent marketing and sexual exploitation. Some other national laws, such as [6,7], require restricting harmful or inappropriate content in various places where access to the Internet is available, for example, in educational institutions and libraries, schools, non-formal education and other child-rearing institutions, as well as in health care institutions, state and municipal institutions.

Although many countries around the world are trying to legally restrict access to information that negatively affects minors, it is quite difficult to enforce these requirements at the state level. Therefore, various technical solutions are used by organizations as well as parents who want to restrict access to harmful or inappropriate content on the Internet themselves. Such solutions can include software, hardware, or both, which are dedicated to filtering web content according to some pre-defined rules. These solutions can be used not only to protect against illegal and harmful online content, but also for other reasons, such as copyright infringement [8]. For example, people often browse the Internet for personal purposes at their workplaces. For this reason, company managers must deal with copyright violations and the use of illegal versions of software on company computers, as well as the leakage of confidential data, cyberattacks, viruses, decreased employee motivation, productivity, and Internet bandwidth.

The available parental control and content filtering solutions are usually implemented as client-side applications, such as Qustodio [9], Norton Family [10], Net Nanny [11], Bark [12], Boomerang [13], etc. The operation of these solutions is based on real-time analysis of the content of visited Internet websites. The administrator can modify the filters used and select certain categories of content that should be blocked or alerted. These categories include, but are not limited to, educational, entertainment, news, sports, shopping, forums, social networks, chat, file sharing, gambling, violence, drugs, weapons, profanity, and mature content. Filtering of the content on the end-user device (client-side filtering) is distinguished by the shortest request processing time. However, as the number of devices increases, this type of filtering becomes ineffective and extremely time-consuming due to the individual configuration and maintenance required for each device. This type of filtering is often used by ordinary people (usually parents) and companies that want to use the filtering system on certain specific devices.

The more flexible and scalable approach is to implement content filtering solutions on a local network, usually on an Internet router, which filters web page requests depending on the predefined filter settings. An Internet content filtering system implemented on

a local network is useful in large organizations, as it is enough to create a single set of content filtering rules and apply it to all network users. The restricted Internet content in different organizations and households, or even in different parts of the same company, may differ depending on various aspects, for example, the age of Internet users or internal rules of the institution that prohibit certain websites. Therefore, it is convenient that in such content filters, it is easy to assign different access levels to certain classes of users. Finally, router-based filtering does not require any additional software to be installed on end-user devices and can be applied in various public places and institutions, as required by the previously mentioned laws.

Currently, most of the commercial (wireless) routers have some kind of parental control and web page filtering firmware installed along with other cybersecurity solutions such as firewalls, MAC (Media Access Control) address filtering, etc. However, parental control and filtering functionality used in routers is quite primitive, based on keywords and simple whitelists and blacklists. These methods ensure good performance and request processing time, but cannot cope with new websites that constantly appear and are not very accurate. Content-based classification of web pages would be more accurate and fully resistant to newly appearing web pages.

Therefore, this study evaluates the possibility of creating a web page classification and filtering solution for common commercial low-resource home routers using machine learning-based (ML-based) text classification methods. Such a solution would not require additional client-side software and would be suitable both for home usage and various public organizations such as schools, libraries, universities, state and municipal institutions, etc.

1.2. Aim and Principal Conclusions

To solve the shortcomings of current web page classification and filtering solutions for home routers, we propose to use text-based web page classification methods applying ML-based classification algorithms, adapted for low computational resource environments.

The aim of this study is to determine whether it is possible to create a text-based web page classification and filtering solution using ML methods, suitable for use in devices of low computational power, such as widely available wireless home routers. To test this hypothesis, implementations of several ML-based text classification algorithms in the C/C++ programming language were created, suitable for deployment in OpenWrt-based embedded systems. The study evaluated these algorithms on a real commercial home router to determine whether machine learning-based web page classification provides sufficient speed and accuracy. The results show that the implementation of the linear SVM (support vector machine) method demonstrates the best accuracy of 0.9198, while the random forest model was the fastest, with 1.5619 s to process a web page.

The main contributions of this paper are listed as follows:

- A text-based web page classification and filtering solution for low-resource home routers applying ML methods was proposed in this study. Compared to current web page detection and filtering methods used on these devices, the proposed system combines the known web page category list approach with ML-based text classification, allowing one to achieve high accuracy, the ability to classify unknown web pages, and a good response time.
- A new dataset (available at [14]) was created for testing purposes. It contains 2648 records with several types of web pages, namely business, education, adult, games, health, sport, and travel.
- Several less resource-demanding implementations of popular ML-based text classification methods were created in the C/C++ programming language, namely softmax regression, decision tree, random forest, and linear SVM. Compared to traditional

text classification methods, we restricted the number of words (tokens) in the feature selection step of the proposed methodology, significantly minimizing the use of computational resources. We proposed selecting only the first N words of each cleaned web page for further processing, which includes stemming and calculation of TF-IDF (term frequency-inverse document frequency) values. We also evaluated how classification accuracy and processing time depend on the size of the features with different N values (200, 150, 100, and 50 words), producing the four different versions of each classification model. The source code for these implementations, including the source code for the creation of the classification models and the source code for the testing of the models on OpenWrt-based devices, is available on the GitHub platform [15].

- An experimental evaluation of the implementations of the proposed ML-based web page classification algorithms was performed in terms of classification accuracy and processing time. The experiments were carried out using a commercial home router Asus RT-AC85P, with 256 MB RAM and a MediaTek MT7621AT 880 MHz CPU, showing good performance and demonstrating the potential to use ML-based algorithms on low-resource devices.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 discusses the methodology of the study, including details on the proposed dataset, implementations of the ML algorithms, and experimental setup. Section 4 presents the results of the evaluation of the implemented ML algorithms on the selected home router. Section 5 discusses the results and evaluates them in the context of previous research. Section 6 summarizes the study and describes future work.

2. Related Work

To protect children or other users from harmful or other unwanted web content, parental control or similar solutions are used, which allow filtering content by restricting access to unwanted Internet resources. As discussed in recent reviews of cyber parental control methods [16,17] various techniques are used for website filtering: browser-based, search engine-based, app-based, and filtering framework-based. These solutions are usually implemented as client-side software, which should be installed and configured by parents or administrators on each user device individually. The authors of [16,17] emphasize that client-side filtering techniques can be easily bypassed using different search engines or browsers, or using proxy and VPN (Virtual Private Network) tools. According to [17], website filtering is usually based-on the URL of the website, keywords, or website content. Although URL-based and keyword-based methods are lightweight and popular among commercial parental control solutions, they are not very effective for several reasons. URL-based methods use a whitelist and a blacklist of websites, which are often incomplete and require constant updates because thousands of new websites emerge every day. Keyword-based methods neglect the context of keywords, leading to over-blocking websites containing any of the selected keywords. Content-based filtering is the most accurate and usually employs ML methods for content classification. But the drawback of this approach is computational complexity and high latency as a consequence [17].

In the scientific literature, the problem of content-based classification (and filtering) of web content has been widely studied for many years, and classification is usually based on text or image analysis, or their combination [17,18]. According to the [18] survey, the most popular ML-based classification methods are SVMs (support vector machines), neural networks and CNNs (convolutional neural networks), kNN (k-nearest neighbors) algorithms, and Naïve Bayes. The most accurate are kNN and Naïve Bayes, which reach an accuracy of more than 99% (using the F1 metric), followed by CNN and SVM [18]. However, the computational cost of the algorithms and methods mentioned above is quite

high, which can be a big obstacle to applying these methods to low-cost, low-computational-power devices, such as wireless home routers. This is especially true for so-called "deep" classification methods, such as CNNs, recurrent neural networks (RNNs), or DLMs (deep language models) as described in the [19] review. For example, the research [20] explores the potential of using large language models (LLMs) as a solution for website categorization and uses ML models that contain more than 70 billion parameters and require multiple GPUs (Graphics Processing Units) to achieve accuracy rates exceeding 70%. Such high requirements cannot be met by any commercial home router. Some authors [21–23] combine multiple text classification methods to produce better classification accuracy. However, the proposed methods are not suitable for the classification of web pages in real-time, since their processing times are too high (e.g., 60–75 s for [22] and 50–200 s for [23]).

Some attempts have been made to optimize content classification methods and make them more applicable in practice. For example, the authors of the article [24] conducted tests to evaluate the accuracy of machine learning models of different sizes for web page classification. They achieved great results by minimizing the original model from 12 GB to 17 MB and sacrificing only around 4% in precision (from 95.70% to 91.41%). However, the authors do not consider the computational speed of the proposed models, which is important for practical applications in devices with low computational power and directly affects the system response time and usability.

Several studies propose practical implementations of parental control and content classification and filtering systems. The authors of [25] present an AI-based child care parental control system. The study proposes a quite complex multilayered server-based system, based on commercial third-party services, that is used to analyze content and activities performed on a child's mobile device. Unlike our approach, additional client-side software should be installed on the child's device, while the backend of the system requires much more resources than an ordinary home router. A similar proxy server principle is also applied in the study [26], which presents the Guardian Eye solution. This solution allows parents to block or filter inappropriate content for their children and consists of three main systems: a proxy server, the parent dashboard, and a mobile application to redirect all outgoing requests to the proxy server [26]. The ChildProtect parental control framework [27] also uses the backend and frontend subsystems. Content analysis is performed on the "sophisticated backend-side," ensuring a deep content analysis of the websites accessed by the child. The most similar study, which mentions the possibility of implementing the proposed content classification solution on the network router device, is [28]. However, the authors of this study mainly evaluate the accuracy of the proposed method, while the characteristics and performance evaluation of the so-called "intelligent wireless router" are not revealed. Therefore, it is unclear whether the proposed solution is suitable for commonly found commercial home routers, which are quite restricted in terms of memory and CPU computational power.

Finally, many commercially available home routers have some kind of parental control and web page filtering functionality. But these filtering and blocking solutions are based on the previously mentioned URL blacklists and whitelists, and keyword-based approaches. These methods are not very effective and accurate, but require fewer resources than more sophisticated ML-based content classification algorithms. Recent research related to the evaluation of parental controls in routers, DNS (Domain Name System) services, and software showed that standard filtering and blocking solutions can block only up to 26–27% of unwanted content [29].

There are many studies suggesting that ML methods are resource-consuming, but until now, there were no studies that evaluated how good or bad these ML methods perform on real-life devices with restricted resources. To our knowledge, no one has yet assessed

whether resource-intensive ML-based text classification methods can be applied directly for web page filtering and blocking in home routers or other low-resource network devices.

3. Materials and Methods

3.1. Model of the Proposed System

The model of the proposed web page classification and filtering system is presented in Figure 1. The central component in this solution is a (wireless) home router, which checks every web page request from connected devices and allows these requests or blocks them, depending on the current *Filtering settings*. The *Filtering settings* are managed by the router administrator, who selects which categories should be shown to the users and which should be blocked. These *Filtering settings* can be changed dynamically at any time without altering the operation of the router. When a user's application requests a web page, the first step is to check if this URL is already known and is in the *Category list of known web pages*. This step is performed by the *Known web pages checking component* and allows us to speed up web page classification, avoiding unnecessary web page analysis for known web pages. If the *Known web pages checking component* cannot find a web page in the list, then it asks for an *ML-based classification component* to classify this page. The *ML-based classification component* downloads the source of the requested web page and classifies it according to the textual content of the page. Then, the *Category list of known web pages* is updated to avoid the time-consuming ML-based classification next time when the user requests the same page. In addition, the *Category list of known web pages* and the *ML model* used for classification can be updated externally, e.g., from the external server of the router manufacturer/provider, further improving the classification performance of the router. The distributed classification approach can also be applied to this model when multiple routers periodically share and synchronize their *Category lists of known web pages*.

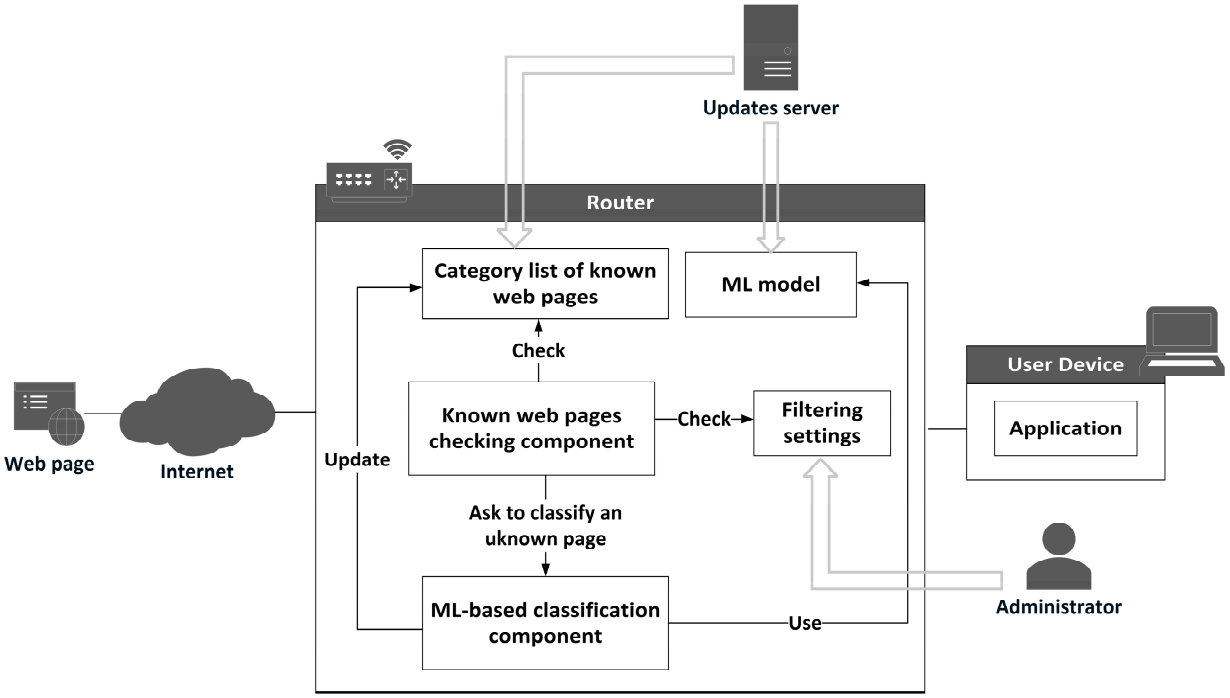

Figure 1. Model of the proposed web page classification and filtering system.

The main goal of this study was to assess whether ML-based textual content classification methods are generally suitable for usage on resource-restricted devices. Therefore, the ML-based classification parts of the proposed system were implemented and experimen-

tally evaluated in this study (the *ML-based classification component* and several *ML models*) since they are the most resource-intensive and affect the performance of the entire system the most. The developed program code includes downloading a web page, processing its text, and assigning it to one of the defined categories.

3.2. Experimental Environment Setup

The software for the experimental evaluation was developed on the Lenovo T16 laptop computer (Windows 11 OS) with 16 GB of RAM, an Intel Core Ultra 5 125U, 1300 MHz, 12 Core(s), 14 Logical Processor(s), and 1TB SSD (solid state disk).

The experimental evaluation was performed using the commercial home router Asus RT-AC85P with 256 MB RAM and MediaTek MT7621AT 880 MHz CPU. The WAN port of the router was connected to the Internet, and all evaluations were performed inside the router by connecting to it using the SSH (Secure Shell) client. The router used the standard OpenWrt firmware v. 24.10.0, which is directly downloadable from the official OpenWrt site. Additional libraries and software were cross-compiled for the mipsel_24kc architecture using the OpenWrt building system. Then, the software, additional libraries, dataset, and pre-trained ML models were downloaded to the router using SSH. Performance evaluation was performed by classifying all records in the test dataset and logging execution times into CSV (Comma-Separated Values) files. The resulting CSV files were downloaded to the PC using SSH and processed using standard spreadsheet software.

3.3. Evaluation Methodology

The experimental evaluation of the performance of the machine learning methods implemented in this study contained two main stages, as shown in Figure 2.

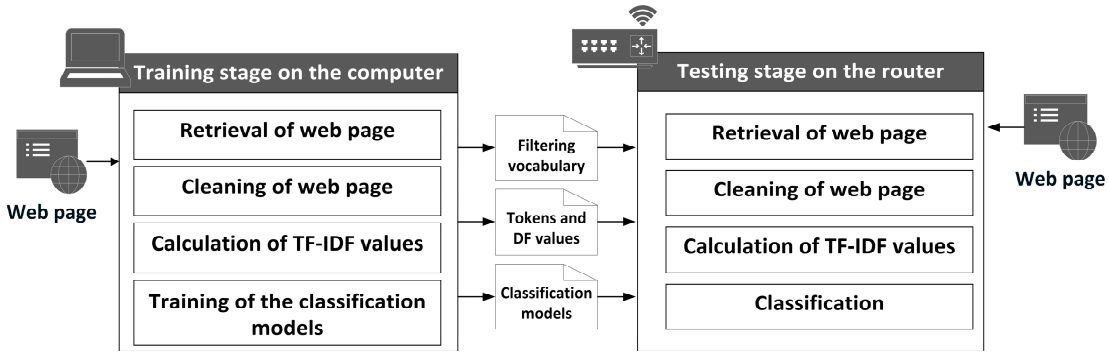

Figure 2. Stages of experimental evaluation.

The first stage was used to train the classification models on the personal computer. The second stage was performed on the real wireless router to evaluate the performance of the trained classification models. For this purpose, we created a custom dataset with seven web page categories, implemented four machine learning-based text classification algorithms in the C/C++ language, which were used to train the classification models on the virtual machine-based Linux platform, and then deployed and evaluated these models and algorithms on the wireless router-based evaluation platform.

The training stage contained the following steps (see Figure 3):

1. The dataset was manually divided into training and testing sets (71% and 29% ratio), to produce sets that are as balanced as possible;
2. For each URL in the training set, a web page HTML (Hypertext Markup Language) source was downloaded and saved for later use to train different ML classification algorithms. Therefore, each classification model was trained using the same data;

3. Each web page source was cleaned, removing script elements, style elements, HTML tags, stop words, and non-English language words;
4. The first *N* words of each web page were selected for further processing (words can repeat in these collections). To evaluate how classification accuracy and processing time depend on the size of the features, several *N* values (200, 150, 100, and 50 words) were used to produce four different versions of each classification model;
5. The filtering dictionary (used later during the testing stage) was produced by combining the previously selected first *N* words of each web page into one collection of non-repeating words;
6. The collections of the first *N* words produced in the fourth step were stemmed, using the Porter Stemming algorithm [30];
7. The DF (document frequency) value of each unique token (stemmed word) from all collections of the first *N* words was calculated and saved to a separate file;
8. The TF (term frequency) value of each unique token of each collection produced in the sixth step was calculated;
9. The TF-IDF (term frequency-inverse document frequency) matrix was created for the training dataset, where the tokens, obtained in the seventh step, are the column headers and the cells contain the TF-IDF weights for each token in each web page (if the token is not presented on that web page, its TF-IDF = 0);
10. Each classification model was trained, using previously calculated TF-IDF values as weights of the tokens used as training features;
11. The trained models were saved in a specific JSON (JavaScript Object Notation) format for deployment to the wireless router.

During the experimental evaluation, we tried to determine how much the classification accuracy and processing time depend on the number of words used as classification features. Therefore, during the training stage, four versions of each classification model (16 classification models in total) were created by selecting different numbers *N* of the first words of the cleaned web page. Steps 4–11 were repeated using different *N* values (200, 150, 100, and 50). This resulted in different versions of classification models with different numbers of features for each *N* value: 16,709 ($N = 200$), 15,127 ($N = 150$), 13,006 ($N = 100$), and 9578 ($N = 50$).

The experimental environment setup contained a few steps:

1. The C/C++ implementations of classification algorithms were compiled, producing binaries which were installed on the wireless router;
2. The dictionary produced in the fifth step of the training stage, together with the DF values file produced in the seventh step of the training stage, and the trained models produced in the eleventh step of the training stage, were deployed to the router;
3. The main evaluation program was installed on the router and was responsible for the preparation of data, execution of the classification algorithms, and measuring the preparation and classification times.

The experimental evaluation test stage (Figure 4) was executed by the main evaluation program as a cycle, where the following steps were repeated for each web page in the test dataset:

1. The HTML source of the web page was downloaded;
2. The source of the web page was cleaned, removing script elements, style elements, and HTML tags;
3. The first 200 words matching the filtering dictionary were selected for further processing (the words could be repeating, and the number of selected words could be less,

but not more than 200), bypassing additional removal of stop words and non-English language words;

4. The stemming was applied to these words, producing tokens;
5. The TF value of each unique token was calculated;
6. The TF-IDF weights were calculated for each token in the DF file, produced previously in the training stage, using the TF values obtained in the fifth step (if the token is not presented in this web page, its TF-IDF = 0);
7. The TF-IDF weights were used as input to the classification models, and the classification results were evaluated.

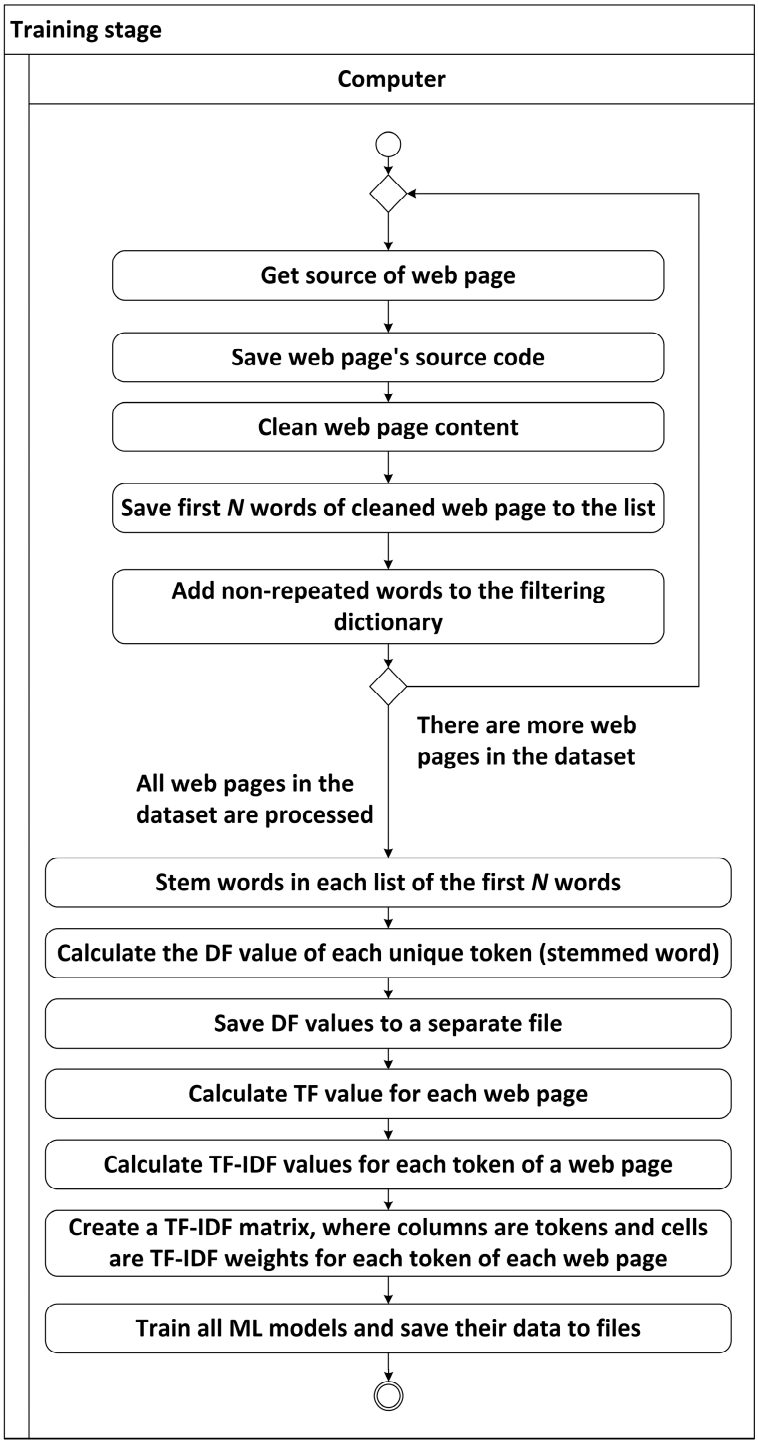

Figure 3. Training stage of experimental evaluation.

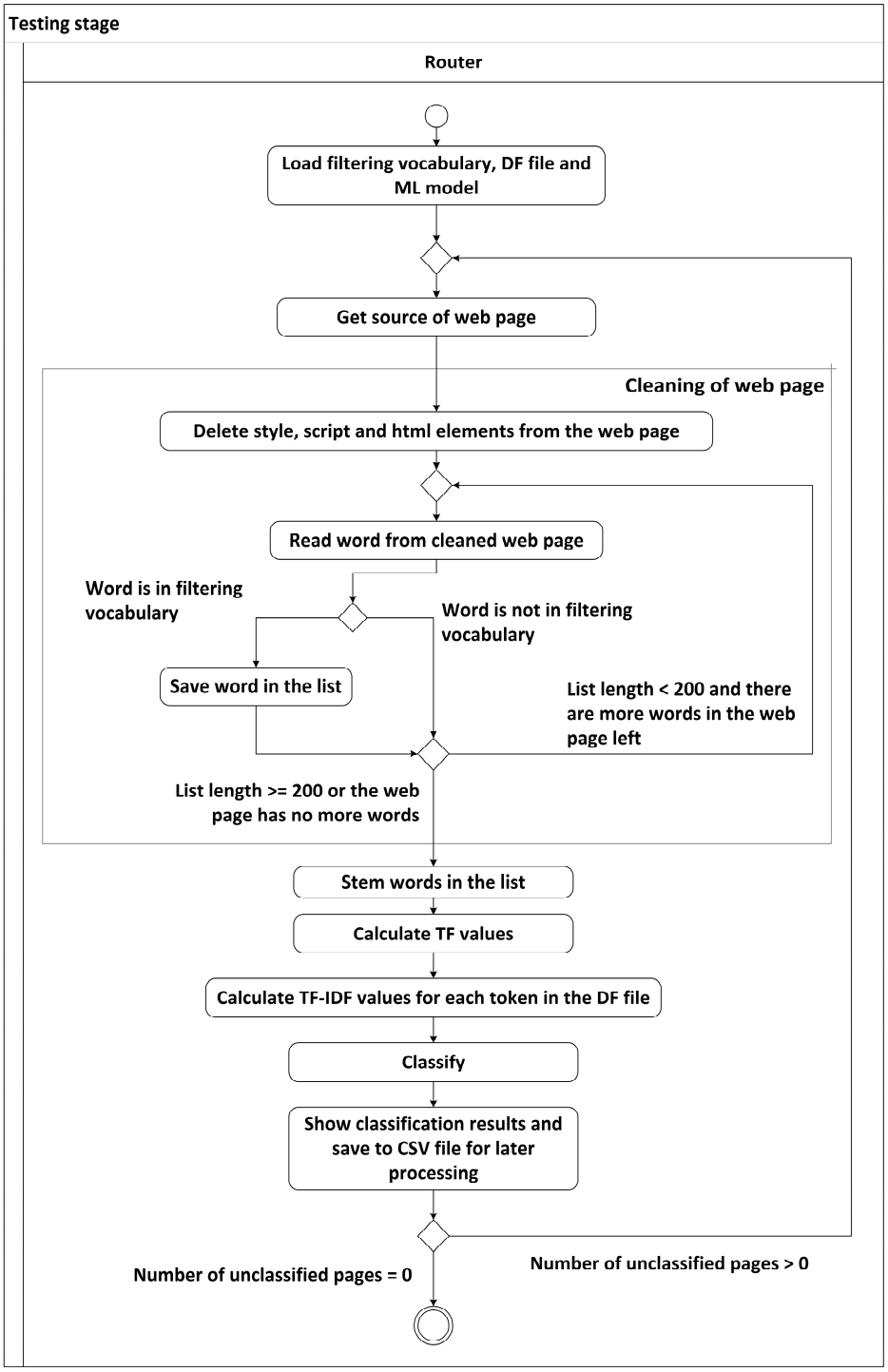

Figure 4. Testing stage of the experimental evaluation.

Each web page in the test dataset was classified once using each of the 16 ML classification models created in the training stage. The classification results and processing times were saved to the CSV files. These files were later processed using spreadsheet software to calculate the average processing times for each ML model, its accuracy-related metrics, and to generate confusion matrices. The *std::chrono::high_resolution_clock* standard C++ class (*now()* function) was used to calculate the processing times of the following experimental evaluation steps: page retrieval time (step 1), page cleaning time (steps 2–4), TF-IDF calculation time (steps 5–6), and classification time (step 7).

3.4. Creation of the Dataset

The custom dataset for this study was created based on the original Curlie dataset [31] by removing non-English web pages, further filtering them by the seven categories selected for our study (the initial dataset contains much more classes), and randomly selecting around 500 pages for each category. Then, we removed the inactive web pages, checked the labels for the remaining ones (since some of them changed their type over time), and manually added additional records of various types. The original Curlie dataset does not contain an adult category, so it was manually added to our dataset. The games category was updated by removing child-oriented web pages and adding gambling-related records.

The resulting dataset contains 2648 records of the following web page types: business, education, adult, games, health, sport, and travel. The created dataset is freely available at [14] for other researchers. A reduction in the original Curlie dataset was performed to minimize web page processing times during the training stage, since processing of 2648 web pages took around 12 h on the laptop computer with 16 GB of RAM and Intel Core Ultra 5 125U CPU.

3.5. Implementation of the Classification Models

The classification models and the evaluation program were implemented using the C/C++ programming language. Although there are many studies that use Python-based implementations of various machine learning algorithms, and it is possible to run Python-based programs on the router, our previous research [32] has shown that Python programs are much slower than those implemented in C/C++. Therefore, we developed our own implementations of all classification algorithms (softmax regression, decision tree, random forest, and linear SVM) used for experimental evaluation. For this purpose, we used the following libraries, namely *curl-4.8.0* [33], *armadillo-14.4.0* [34], *cereal-1.3.0* [35], *ensmallen-2.22.1* [36], *mlpack-4.6.2* [37], and the C++ implementation of the Porter Stemming algorithm [30]. The parameters of the classification algorithms are presented in Table 1.

Table 1. Parameters of the classification algorithms, depending on the number of first words of the web pages used for classification.

Classification Algorithm	Number of First Words of Web Pages Used for Classification	Parameters of Classification Algorithm
Softmax regression	200, 150, 100, 50	lambda = 0.0001
Linear SVM	200, 150, 100, 50	lambda = 0.00005, delta = 0.01
Decision tree	200	minLeafSize = 10, maxDepth = 0, minGainSplit = 1×10^{-7}
	150	minLeafSize = 7, maxDepth = 0, minGainSplit = 1×10^{-7}
	100	minLeafSize = 6, maxDepth = 0, minGainSplit = 1×10^{-7}
	50	minLeafSize = 5, maxDepth = 0, minGainSplit = 1×10^{-7}
Random forest	200	numTrees = 25, minLeafSize = 10, maxDepth = 0, minGainSplit = 1×10^{-7}
	150	numTrees = 20, minLeafSize = 10, maxDepth = 0, minGainSplit = 1×10^{-7}
	100	numTrees = 20, minLeafSize = 5, maxDepth = 0, minGainSplit = 1×10^{-7}
	50	numTrees = 20, minLeafSize = 5, maxDepth = 0, minGainSplit = 1×10^{-7}

As described in Section 3.3, there are two stages in our methodology: training and experimental evaluation (testing on the real router). We created two different programs that follow the steps of these stages. The training program was run on a desktop computer

and used to create classification models. These models were later deployed on the router and used by the evaluation program to calculate the performance values of these models. The main difference between the training and the evaluation programs is that the training program uses regex statements for web page source cleaning and an English dictionary to remove non-English words. Since regex statements are very resource-intensive, they are not suitable for use on the router. For that reason, the custom HTML element search and removal algorithm was implemented to clean the web page source. This algorithm first removes all <script> elements, then removes all <style> elements, and finally removes all remaining tags (searching for < and > characters), leaving the inside text for further processing.

The software was developed using MS Visual Studio 2022 (v143) with the following parameters: C++ language standard—ISO C++ 17 standard, Windows SDK version—10.0, compiler v14.44.35207. The source code for the router was built using the official OpenWrt building system for the mipsel_24kc architecture. The developed source code is available on the GitHub platform [15].

4. Results

The summarized results of the experimental evaluation are presented in Table 2. This table presents both the classification metrics of the algorithms evaluated and the average computation times of these algorithms. The classification metrics include accuracy, recall, and F1-score, and the average processing time consists of the average times of downloading the web page, cleaning it, calculating the TF-IDF values, and classifying. Each of the evaluated classifiers was tested with different numbers N of words used for classification to determine the minimum number of words that would guarantee sufficient classification accuracy. This is because a larger number of words leads to longer computations, a larger number of features, and the amount of memory used.

Table 2. Performance of the classification algorithms, depending on the number of first words of the web pages used for classification.

N *	Classification Model	Classification Metrics			Average Processing Time (Milliseconds)				
		Accuracy	Recall	F1-Score	Web Page Retrieving	Web Page Cleaning	TF-IDF Calculation and Data Preparation	Classification	Total
200	Softmax regression	0.9085	0.8978	0.9023	1014.8	94.6	643.5	52.3	1806.8
	Linear SVM	**0.9198**	**0.9179**	**0.9201**	1073.0	96.0	638.7	52.1	1861.1
	Decision tree	0.6989	0.6842	0.6974	1010.9	97.3	615.3	**33.1**	1758.2
	Random forest	0.7903	0.7825	0.7743	978.8	96.0	642.2	34.0	**1752.4**
150	Softmax regression	0.9057	0.8949	0.8991	1005.6	95.5	588.4	47.5	1738.6
	Linear SVM	**0.9183**	**0.9146**	**0.9181**	1056.1	93.3	590.1	47.2	1788.0
	Decision tree	0.7244	0.7119	0.7176	1004.4	94.3	570.5	**30.0**	1700.8
	Random forest	0.7717	0.7608	0.7589	983.7	95.6	575.7	30.5	**1687.0**
100	Softmax regression	0.9055	0.8894	0.8948	1028.7	95.2	506.1	41.0	1672.3
	Linear SVM	**0.9111**	**0.9068**	**0.9097**	1034.8	94.8	510.7	40.0	1681.7
	Decision tree	0.7093	0.6998	0.7032	1022.5	94.4	506.3	**26.0**	1650.5
	Random forest	0.7931	0.7867	0.7844	975.2	96.4	528.1	26.3	**1627.6**
50	Softmax regression	0.8912	0.8731	0.8785	1107.6	94.9	418.7	30.1	1652.6
	Linear SVM	**0.9066**	**0.8985**	**0.9024**	1115.6	94.8	402.8	29.1	1643.9
	Decision tree	0.6819	0.6679	0.6659	1065.5	96.5	419.9	**19.0**	1602.4
	Random forest	0.7879	0.7788	0.7786	1040.2	95.0	405.5	19.9	**1561.9**

* Number of first words of web pages used for classification.

As we can see in Table 2, the best accuracy was demonstrated by the linear SVM classifier, which was the most accurate with all N words selected for classification. In terms of accuracy, softmax regression was not far behind. Meanwhile, the decision tree algorithm performed the worst, with an average accuracy lower by 0.21. Since it is more important to recognize as many instances of a category as possible when classifying and blocking web pages, the recall metric is one of the most important indicators of the quality of the algorithm. In the case of the problem we are solving, it is better to assign a page to a prohibited category (e.g., adult, games) even if it does not belong to it than to fail to recognize a truly harmful page. As presented in Table 2, linear SVM also has the best recall value, reaching 0.9179 when N = 200.

As shown in Table 2, the total average processing times for all algorithms are quite similar, where the difference between the fastest and the slowest classifiers is within 0.11 s. The user would not notice this difference in real-life applications. Therefore, it is better to use the linear SVM classifier due to its accuracy. The more detailed results of the linear SVM classification algorithm for each combination of N first words of web pages used for classification are presented in Table 3. As we can see in Table 3, the best classification precision was achieved for the adult category (1.0) for each value of N. The second-best classification precision was obtained for the sport category (when N = 200, 150, 50), except for the case of N = 100, where the travel category had a better classification precision. However, the adult and sport categories had the worst recall scores with all N values, which means that some pages in these categories were classified as other categories.

Table 3. Classification results of the linear SVM classification algorithm depending on the number N of first words of web pages used for classification.

Web Page Category	Number of First Words of Web Pages Used for Classification							
	N = 200		N = 150		N = 100		N = 50	
	Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0—Business	0.9098	0.9098	0.8971	0.9173	0.9160	0.9160	0.9313	0.9173
1—Education	0.8208	0.9063	0.8396	0.9175	0.8350	0.9053	0.8349	0.9479
2—Adult	**1.0000**	0.8793	**1.0000**	0.8571	**1.0000**	0.8421	**1.0000**	0.7857
3—Games	0.9099	**1.0000**	0.9174	**0.9901**	0.8818	**0.9798**	0.8761	**0.9900**
4—Health	0.9048	0.9135	0.9048	0.9135	0.8879	0.9135	0.8942	0.8942
5—Sport	0.9697	0.8421	0.9596	0.8407	0.9300	0.8158	0.9444	0.7798
6—Travel	0.9661	0.9744	0.9580	0.9661	0.9669	0.9750	0.9206	0.9748
Average:	**0.9259**	**0.9179**	**0.9252**	**0.9146**	**0.9168**	**0.9068**	**0.9145**	**0.8985**

As shown in the confusion matrixes (see Figure 5), some of the adult pages were incorrectly assigned to the games, sport, and business categories. The reason for this could be the absence of some very specific words used in adult web pages, so the classifier assigned these pages to other categories. Additionally, adult web pages usually contain more images than text, making their classification more difficult using text-based methods. Also, it should be considered that the adult category contained the least number of web pages due to the constantly changing and disappearing URLs of such web pages.

However, the adult category had perfect precision, which means that no other pages were wrongly assigned to this class. Some of the sport category web pages were incorrectly assigned to all other categories except for the adult category. Independent of the N number, most of the incorrectly classified pages of the sport category were assigned to the education category, possibly because the fields of sport and education are quite related. The best recall was achieved for the games category, followed by travel for every N value. As shown in Figure 5, only a few pages in the game category were incorrectly assigned to the sport and

health categories, while some of the travel pages were assigned to the business, sport, and games categories.

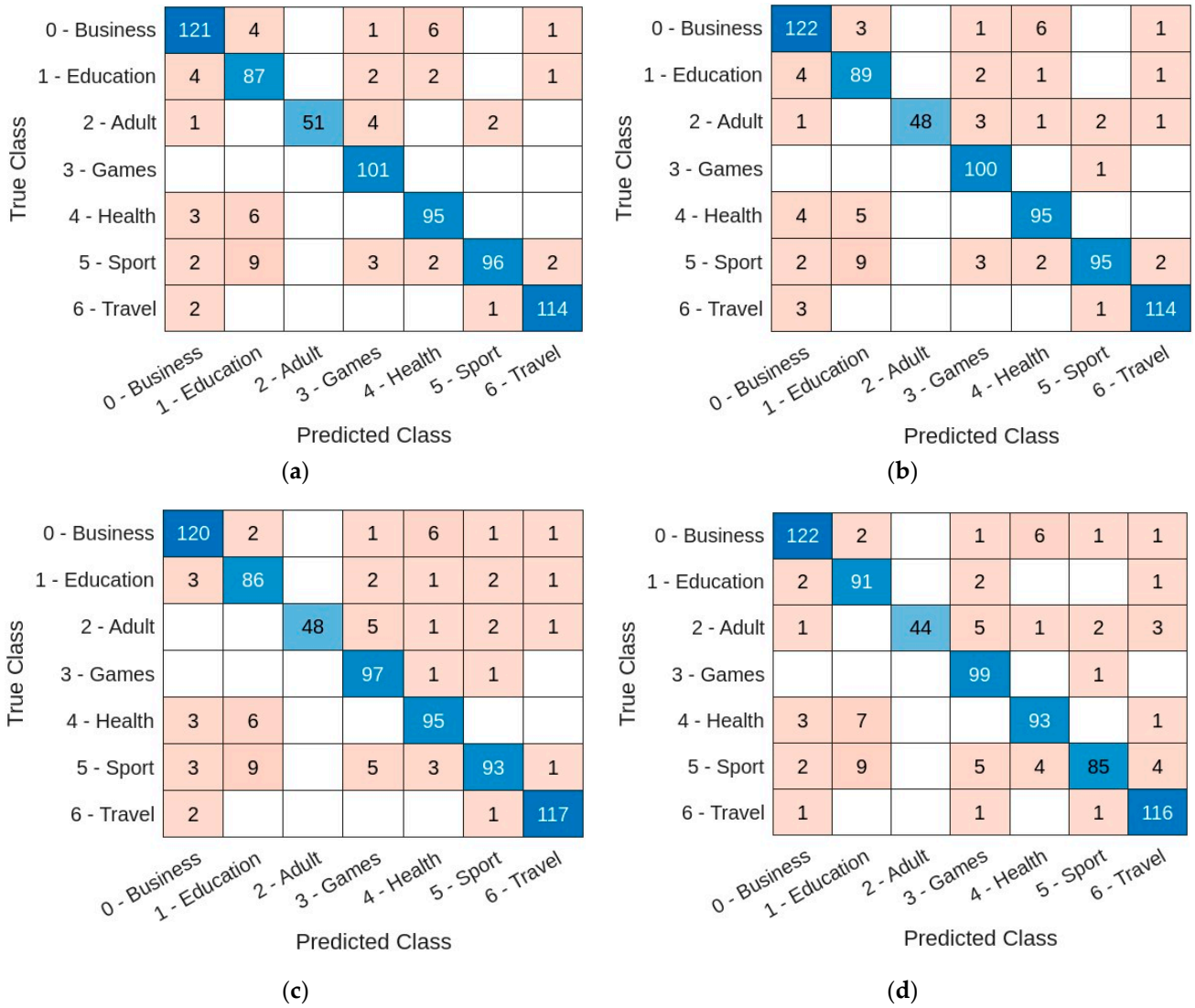

Figure 5. Confusion matrices of the linear SVM classification algorithm using N first words of web pages: (**a**) N = 200; (**b**) N = 150; (**c**) N = 100; and (**d**) N = 50.

Therefore, the results obtained show that though the N number of the first web page words used for classification has an impact on the classification quality, the differences in accuracy, precision, recall, and F1-score are not very significant. Decreasing the N number leads to a decrease in all classification metrics, but the difference between the best and the worst accuracy is less than 2%. For example, this difference for the linear SVM classifier is only 0.0132. However, the smaller number N makes the calculations faster, and the difference between the fastest and slowest average processing time using the linear SVM classifier is 0.2172 s (a decrease of 12%).

The linear SVM algorithm with N = 50 was used to evaluate the possible impact of the classification of concurrent HTTP requests on the average classification time. The first 400 rows of the dataset were separated, additionally divided into two and four equal parts, and used for this experiment. Each variant of the evaluation data was placed into the router and processed using one, two, and four parallel processes that processed 400, 200, and 100 records, respectively. The evaluation results are summarized in Table 4.

Table 4. Scalability assessment of the linear SVM classification algorithm (N = 50), depending on the number of parallel processes used.

Number of Parallel Processes	Average Processing Time (Milliseconds)					Overhead, %	Total Time to Process All Records, s
	Web Page Retrieving	Web Page Cleaning	TF-IDF Calculation and Data Preparation	Classification	Total		
1	996.32	92.22	396.29	27.28	1513.63	0.00	609
2	1085.12	102.56	418.62	32.27	1640.03	8.35	333
4	1152.07	139.23	561.42	44.61	1898.78	25.45	202

The results show that parallel processing is more efficient in terms of the total time required to process all records. If four parallel processes are used, then it requires three times less time to classify all URLs (202 s vs. 609 s). On the other hand, the classification time of a single URL increases when the router is processing several web page requests simultaneously. The good thing is that this increase is only about 25% when three additional processes run similar classification tasks.

Another interesting aspect of the resource efficiency evaluation of the proposed approach is the RAM requirements. Home routers usually have limited RAM, which could become one of the main limiting factors. On the other hand, it is not a trivial task to evaluate the real RAM requirements, because RAM is used not only for running processes but also is shared by RAM-based file systems, which are used for temporary file storage, logging, etc. Therefore, only some basic observations obtained using the "free" utility built into the OS could be provided. The memory available after a clean boot of the router OS reported by the "free" utility was 154 MB (from a total of 256 MB). After uploading all required additional libraries, software, and ML model, the available memory decreased to 118 MB. This is a consequence of the fact that the router's file system uses RAM to store data. The three largest objects are as follows: the openBLAS library (~15 MB), classification software (~2 MB), and pre-trained ML models (which may be several MB, depending on the format used and the type of model). The available memory does not decrease dramatically when one, two, or four classification processes are running (112 MB, 111 MB, and 105 MB, respectively). This is the expected behavior, as in modern OS, only private data uses additional RAM when several similar processes are running.

5. Discussion

This study appears to be the first to evaluate whether commonly available low-resource wireless routers can run text-based ML web page classification algorithms with sufficient accuracy and processing speed. As presented in Section 4, the best classification accuracy was achieved using the linear SVM classifier, reaching the accuracy of 0.9198, the recall of 0.9179, and the F1-score of 0.9201. These results are in line with previous studies [18], which confirms that the SVM classification algorithm is among the most accurate.

Compared to the study [24] results, which proposes three small-scale fastText machine learning models with memory footprints of 154 MB, 47 MB, and 3 MB, we achieved slightly better F1-score (0.90–0.91 vs. 0.92). However, we focused on computational speed, while the authors of [24] tried to minimize memory usage, sacrificing accuracy.

Compared to the study [28], which proposes an automatic detection system for porn and gambling websites using image-based and text-based classification, our evaluation showed worse results. We achieved the precision of 1.0 and the recall of 0.8793 for the adult category, and the precision of 0.9099 and the recall of 1.0000 for the game category. The authors of [28] achieved the precision of 0.9971 and the recall of 0.9886 for the porn category, and the precision of 0.9942 and the recall of 0.9884 for the gambling category. However,

it is difficult to directly compare our results with [28], because we used only text-based classification algorithms. The longest detection time of the system proposed in [28] did not exceed 1.157 s, while the average detection time was within 1 s. The longest average detection time in our study was 1.8611 s (linear SVM classifier, $N = 200$), and the shortest average detection time was 1.6024 s (decision tree classifier, $N = 50$). However, the authors of [28], while mentioning that their system can be applied to the intelligent wireless router, did not provide any specification of the hardware and software that were used for the study. Therefore, it is not clear whether the traditional ML algorithms evaluated in our study would be slower than the system proposed in [28] deployed on the same low-resource wireless home router.

In summary, our study has shown that it is possible to achieve higher than 0.9 web page classification accuracy using common text-based ML classification algorithms running on the widespread low-resource wireless home routers. Previous studies suggest that the accuracy can reach even 0.99, but more complicated and resource-intensive classification algorithms should be used. This would lead to a much longer average classification time, which in our case was 1.8611 s for the linear SVM-based algorithm. Furthermore, it is necessary to remember that we proposed a less resource-demanding implementation of popular ML-based text classification methods, using only the first N words of cleaned web pages to produce classification models. This approach was used to shorten the training times for the ML models and the processing times of the classification programs running on the router. As we can see in Tables 2 and 3, increasing the number of words N used for classification increases the accuracy of the classification. However, the calculation time also increases even more. The percentage difference between the accuracy when $N = 50$ and $N = 200$ is less than 2%, while the difference in calculation time is greater than 12%. As we can see from Table 2, the largest part of the calculation is the TF-IDF calculation and data preparation. Therefore, it is obvious that without limiting the size of N and using all the words of the cleaned web page for the preparation of the model data, the total classification time would increase several times.

Regarding the average classification time of 1.8611 s of the linear SVM method, it should be noted that in the proposed system, only newly opened pages that are not on the category list of known web pages will be classified. Therefore, a greater delay will only be felt when opening a page of an unknown website for the first time. Most users, especially those with a slower Internet connection, would barely notice this delay. Furthermore, according to Google PageSpeed Insights [38], which presents data on real user experience during the previous 28-day collection period, the good page load time falls within 2.5 s. Page loading times between 2.5 and 4 s need improvement, while loading times greater than 4 s are considered poor. Even in those rare cases, when multiple user requests should be processed simultaneously, the proposed approach does not heavily affect the response time. The experiments showed that simultaneous classification of four webpages added only 25% to the average processing time.

Thus, when evaluating our measured average classification times, we can state that they are quite good considering the limited resources of the router used in the experiment. It should also be noted that the largest part of the time is not even the calculations related to classification, but simply downloading the page to the router, which is influenced by various factors, such as network speed, quality of the Internet infrastructure, additional restrictions and protections of Internet service providers, etc. Thus, it is very likely that this part of the calculations may differ significantly under certain conditions.

A downside of the proposed approach used is that it deals with text-based content only. Such a solution is not applicable to categorize other types of web content, such as audio, video, images, etc. Also, the proposed solution on its own is not fully resistant to the

cases where users try to intentionally bypass the filter, using various means, such as VPN, SSL (Secure Socket Layer) proxy, etc. However, the router-based solution is more difficult to bypass than the commonly available commercial client app-based filters.

The results presented in this study show high potential in using ML-based text classification and filtering methods for low-resource home routers or other network devices, used for harmful and inappropriate web page blocking. The advantage of router-based filtering is the elimination of additional client-side blocking software, better resilience to circumvention by ordinary users, as well as high flexibility and adaptability for different user groups on the local network. To make such an approach even more appealing for real-life applications, further research can be conducted to optimize web page processing algorithms and to shorten processing times even more.

6. Conclusions

This paper presents the practical implementation of text-based web page classification and filtering algorithms using ML classifiers and their experimental evaluation on the low-resource wireless home router. The aim of this study was to find out whether ML-based classification algorithms can be used on the widely available low-resource wireless home routers, providing sufficient accuracy and still acceptable processing time. The C/C++ implementations of softmax regression, decision tree, random forest, and linear SVM classification algorithms were created and evaluated on the commercial home router.

The results of this study show that it is possible to apply ML-based web page classification and filtering methods on low-resource network devices, achieving a classification accuracy of more than 91% with a processing time of less than 1.9 s and a classification accuracy of more than 90% with a processing time of less than 1.7 s (using linear SVM classifier).

Therefore, the results presented in this study show high potential in applying ML-based text classification and filtering methods for commonly available inexpensive home routers, eliminating the additional client-side blocking app software and increasing resilience to attempts to circumvent web page filtering and blocking solutions by their users.

We suggest that future research should examine the possibility of further optimization of the web page retrieving and processing algorithms, to decrease total processing time and make such a solution more convenient for end-users and more accurate, since the accuracy is directly related to the complexity of the algorithms and their processing time. Furthermore, additional research should be conducted to investigate the influence of a more balanced training dataset on the performance of the proposed system.

Author Contributions: Conceptualization, A.J. and A.L.; methodology, A.J. and A.L.; software, A.J. and N.M.; validation, A.J., A.L. and N.M.; formal analysis, A.L. and N.M.; investigation, A.J. and N.M.; resources, A.L.; data curation, A.J.; writing—original draft preparation, A.J. and A.L.; writing—review and editing, A.J., A.L. and N.M.; visualization, A.J.; supervision, A.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in this study are openly available in Zenodo repository at https://www.doi.org/10.5281/ZENODO.15828692 (accessed on 8 July 2025). The original source code presented in the study is openly available in GitHub repository at https://github.com/ktu-wcc/wcc.git (accessed on 6 August 2025).

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

References

1. Petrosyan, A. Internet Usage Worldwide—Statistics & Facts. Available online: https://www.statista.com/topics/1145/internet-usage-worldwide/ (accessed on 5 August 2025).
2. Kemp, S. Digital 2025: Global Overview Report. Available online: https://datareportal.com/reports/digital-2025-global-overview-report (accessed on 5 August 2025).
3. Unicef. Keeping Children Safe Online. Available online: https://www.unicef.org/protection/keeping-children-safe-online (accessed on 3 July 2025).
4. Federal Communications Commission. Children's Internet Protection Act (CIPA). Available online: https://www.fcc.gov/consumers/guides/childrens-internet-protection-act (accessed on 3 July 2025).
5. Niestadt, M. Protecting Children Online: Selected EU, National and Regional Laws and Initiatives. Available online: https://www.europarl.europa.eu/thinktank/en/document/EPRS_BRI(2025)769570 (accessed on 3 July 2025).
6. Seimas of the Republic of Lithuania. Law on the Protection of Minors Against Detrimental Effect of Public Information. 10 September 2002. No. IX–1067. Available online: https://e-seimas.lrs.lt/portal/legalAct/lt/TAD/TAIS.216702 (accessed on 3 July 2025).
7. Seimas of the Republic of Lithuania. Gaming Law. 17 May 2001. No. IX-325. Available online: https://e-seimas.lrs.lt/portal/legalAct/lt/TAD/TAIS.347707 (accessed on 7 August 2025).
8. Rojszczak, M. Online Content Filtering in EU Law—A Coherent Framework or Jigsaw Puzzle? *Comput. Law Secur. Rev.* **2022**, *47*, 105739. [CrossRef]
9. Qustodio Parental Control App. Available online: https://www.qustodio.com/en/ (accessed on 4 August 2025).
10. Norton Family. Available online: https://us.norton.com/products/norton-family (accessed on 4 August 2025).
11. Net Nany. Available online: https://www.netnanny.com (accessed on 3 July 2025).
12. Bark-Parental Controls for Families. Available online: https://www.bark.us/ (accessed on 4 August 2025).
13. Boomerang Parental Control. Available online: https://useboomerang.com/ (accessed on 4 August 2025).
14. Janaviciute, A.; Liutkevicius, A.; Morkevicius, N. Dataset for Web Page Classification. Available online: https://zenodo.org/doi/10.5281/zenodo.15828692 (accessed on 7 August 2025).
15. Janaviciute, A.; Morkevicius, N. Web Page Classification Software. Available online: https://github.com/ktu-wcc/wcc (accessed on 7 August 2025).
16. Altarturi, H.H.M.; Saadoon, M.; Anuar, N.B. Cyber Parental Control: A Bibliometric Study. *Child. Youth Serv. Rev.* **2020**, *116*, 105134. [CrossRef]
17. Altarturi, H.H.M.; Badrul Anuar, N. A Preliminary Study of Cyber Parental Control and Its Methods. In Proceedings of the 2020 IEEE Conference on Application, Information and Network Security (AINS), Kota Kinabalu, Malaysia, 17 November 2020; pp. 53–57.
18. Hashemi, M. Web Page Classification: A Survey of Perspectives, Gaps, and Future Directions. *Multimed. Tools Appl.* **2020**, *79*, 11921–11945. [CrossRef]
19. Gasparetto, A.; Marcuzzo, M.; Zangari, A.; Albarelli, A. A Survey on Text Classification Algorithms: From Text to Predictions. *Information* **2022**, *13*, 83. [CrossRef]
20. Sava, D. Text-Based Classification of Websites Using Self-Hosted Large Language Models: An Accuracy and Efficiency Analysis. Bachelor's Thesis, University of Twente, Enschede, The Netherlands, 2024.
21. Shin, G.-Y.; Jang, Y.; Kim, D.-W.; Park, S.; Park, A.-R.; Kim, Y.; Han, M.-M. Dark Side of the Web: Dark Web Classification Based on TextCNN and Topic Modeling Weight. *IEEE Access* **2024**, *12*, 36361–36371. [CrossRef]
22. Sun, G.; Zhang, Z.; Cheng, Y.; Chai, T. Adaptive Segmented Webpage Text Based Malicious Website Detection. *Comput. Netw.* **2022**, *216*, 109236. [CrossRef]
23. Nazari, M.; Sadr, H.; Rostami, S.; Khodaverdian, Z. Enhancing Web Page Classification Through a Semantic-Aware and Efficient Focused Crawling Methodology. In Proceedings of the 2025 11th International Conference on Web Research (ICWR), Tehran, Iran, 16–17 April 2025; pp. 169–174.
24. Sobrier, M. Efficient Website Classification: Using Small-Scale Machine Learning Models for a Broad Range of Categories. In Proceedings of the 2024 IEEE International Conference on Future Machine Learning and Data Science (FMLDS), Sydney, Australia, 20 November 2024; pp. 252–257.
25. Jayasekara, U.; Maniyangama, H.; Vithana, K.; Weerasinghe, T.; Wijekoon, J.; Panchendrarajan, R. AI-Based Child Care Parental Control System. In Proceedings of the 2022 4th International Conference on Advancements in Computing (ICAC), Colombo, Sri Lank, 9–10 December 2022; pp. 120–125.
26. Joshi, K.; Trivedi, R.; Patel, J.; Karani, R. Guardian Eye: Hoslitic Parental Control Solution. In Proceedings of the 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 24 June 2024; pp. 1–7.

27. Ameur, H.; Rekik, A.; Jamoussi, S.; Hamadou, A.B. ChildProtect: A Parental Control Application for Tracking Hostile Surfing Content. *Entertain. Comput.* **2023**, *44*, 100517. [CrossRef]
28. Chen, Y.; Zheng, R.; Zhou, A.; Liao, S.; Liu, L. Automatic Detection of Pornographic and Gambling Websites Based on Visual and Textual Content Using a Decision Mechanism. *Sensors* **2020**, *20*, 3989. [CrossRef] [PubMed]
29. Liberato, M.; Affinito, A.; Meijerink, B.; Jonker, M.; Botta, A.; Sperotto, A. To Block Or Not To Block?: Evaluating Parental Controls Across Routers, DNS Services, and Software. In Proceedings of the Network Traffic Measurement and Analysis Conference 2025 (TMA'2025), Copenhagen, Denmark, 10–13 June 2025; International Federation for Information Processing (IFIP): Laxenburg, Austria, 2025.
30. Upadhyaya, R. Porter Stemming Algorithm: C++ Implementation. Available online: https://github.com/rashup/Porters-Stemming-Algorithm/blob/master/PorterStemming.cpp (accessed on 3 July 2025).
31. Lugeon, S.; Piccardi, T. Curlie Dataset—Language-Agnostic Website Embedding and Classification. Available online: https://figshare.com/articles/dataset/Curlie_Dataset_-_Language-agnostic_Website_Embedding_and_Classification/19406693/5 (accessed on 4 July 2025).
32. Plauska, I.; Liutkevičius, A.; Janavičiūtė, A. Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller. *Electronics* **2022**, *12*, 143. [CrossRef]
33. Stenberg, D. CURL: Command Line Tool and Library for Transferring Data with URLs. Available online: https://github.com/curl/curl (accessed on 3 July 2025).
34. Sanderson, C.; Curtin, R. Armadillo: An Efficient Framework for Numerical Linear Algebra. In Proceedings of the 2025 17th International Conference on Computer and Automation Engineering (ICCAE), Perth, Australia, 20–22 March 2025; pp. 303–307.
35. Grant, W.S.; Randolph Voorhies. Cereal-A C++11 Library for Serialization. Available online: https://uscilab.github.io/cereal/ (accessed on 3 July 2025).
36. Curtin, R.R.; Edel, M.; Prabhu, R.G.; Basak, S.; Lou, Z.; Sanderson, C. The Ensmallen Library for Flexible Numerical Optimization. *J. Mach. Learn. Res.* **2021**, *22*, 1–6.
37. Curtin, R.R.; Edel, M.; Shrit, O.; Agrawal, S.; Basak, S.; Balamuta, J.J.; Birmingham, R.; Dutt, K.; Eddelbuettel, D.; Garg, R.; et al. Mlpack 4: A Fast, Header-Only C++ Machine Learning Library. *J. Open Source Softw.* **2023**, *8*, 5026. [CrossRef]
38. About PageSpeed Insights. Google for Developers Newsletter. Available online: https://developers.google.com/speed/docs/insights/v5/about (accessed on 29 July 2025).