WEB CRAWLER: DESIGN AND IMPLEMENTATION FOR EXTRACTING ARTICLE-LIKE CONTENTS

Ngo Le Huy Hien
Department of Sciences and Technologies
University of Lorraine
France
ngo-le-huy-hien7@etu.univ-lorraine.fr

Thai Quang Tien
Est Rouge Technologies JSC
Vietnam
tientq@tech.est-rouge.com

Nguyen Van Hieu*
Department of Information and Technology
University of Science and Technology, The University of Danang
Vietnam
nvhieuqt@dut.udn.vn

Article history:
Received 04.10.2020, Accepted 20.11.2020

Abstract

The World Wide Web is a large, wealthy, and accessible information system whose users are increasing rapidly nowadays. To retrieve information from the web as per users' requests, search engines are built to access web pages. As search engine systems play a significant role in cybernetics, telecommunication, and physics, many efforts were made to enhance their capacity. However, most of the data contained on the web are unmanaged, making it impossible to access the entire network at once by current search engine system mechanisms. Web Crawler, therefore, is a critical part of search engines to navigate and download full texts of the web pages. Web crawlers may also be applied to detect missing links and for community detection in complex networks and cybernetic systems. However, template-based crawling techniques could not handle the layout diversity of objects from web pages. In this paper, a web crawler module was designed and implemented, attempted to extract article-like contents from 495 websites. It uses a machine learning approach with visual cues, trivial HTML, and text-based features to filter out clutters. The outcomes are promising for extracting article-like contents from websites, contributing to the search engine systems development and future research gears towards proposing higher performance systems.

Key words

Web Crawler, Search Engine System, Crawling Mechanism, Machine Learning, Support Vector Machine, Grid Search, Cross Validation.

1 Introduction

While the World Wide Web (commonly known as the Web) comprises a tremendous amount of information from different areas, its content structure is not centrally organized in a specified way and has no predefined data model. [Mini and Jatinder, 2014] The data presented in the Web normally contains more text data which could have various dissimilar formats. [Jain and Subodh, 2018] A Web crawler is invented as a computer program to download data from the World Wide Web in a systematic, methodical, and automated manner. [Avinash et al., 2010; Kausar et al., 2013] It is also named as a spider or a spider-bot, ant, automatic indexer, bot, worm [Kobayashi and Takeda, 2000], and is typically used for Web indexing.

The World Wide Web has a graphical structure in which links displayed on a web page could be used to open other web pages. The Internet can be described as a directed graph, consisting of webpages (nodes) and hyperlinks (edges). Therefore, it can be summarized that the search operation is a traversing process of the directed graph (the Internet). [Kausar et al., 2013] Using the graphical structure of the World Wide Web, web crawlers can move from page to page and traverse some new web pages from a web page. From there, the process of web crawlers starts from retrieving web pages, then inserting them into local repositories [Martin et al., 2004]. As a consequence, web crawlers generate a replica of all visited pages which later be processed and indexed by search engines. [Kausar et al., 2013; Pant et al., 2004] Search engines may store information about web pages that are retrieved from the World Wide Web, making the searching process fast, accurate, and productive [Bruce et al., 2009].

Generally, the purpose of web crawlers is to collect, maintain, manage the index of web pages, and keep the database in the repositories up-to-date [Jain and Subodh, 2018]. With a swiftly growing Internet, web crawling hence is an essential technique to collect data and keep up with web data for search engine systems. Web crawlers can also be applied for automatic maintenance for websites, including validating links or verifying HTML code. [Komal and Ashutosh, 2016] Moreover, some may use them to collect particular kinds of information from webpages, like gathering e-mail addresses, typically for spam [Desai, 2017].

2 Literature Review

"WebCrawler" was first launched in 1994, with the world's first full-text crawler and search engine [Pinkerton, 1994]. The "WebCrawler" allowed users to search web document contents rather than keywords and descriptors, lessening confusing results, and providing better search capacities. Since then, there are numerous researchers have used and studied web crawlers and crawling techniques, which are valuable in examining the current work.

Mridul B. Sahu et al. [Mridul and Bharne, 2016], for example, revealed in their paper two main approaches to control the crawler decisions, the supervised learning approach, and the traversing and ranking all links approach. They proposed that during the learning process, a web crawler may deliver intelligent decisions on choosing its strategy. A basic web crawling technique and architecture can be seen from the research of Christopher Olston and Marc Najork [Christopher and Marc, 2010]. Vladislav Shkapenyuk and Torsten Suel [Vladislav and Torsten, 2001] have designed and implemented a distributed web crawler in their paper. Also presenting a distributed web crawler, Dhiraj Khurana, and Satish Kumar [Dhiraj and Satish, 2012] described its pros and cons, along with other types of web crawler. Meanwhile, different crawling technologies and techniques for crawling hidden web documents were proposed in a research paper of Beena Mahar and C K Jha. [BeenaMahar and Jha, 2015] Introducing by Priyanka-Saxena [Priyanka, 2019], a scalable web crawler was written in java called Mercator, becoming a commercial use source.

Some studies have been instituted for applying in specific areas. For example, Minas Gjoka [Minas, 2010] used a web crawler to extract data for the statistical properties of online social networks. Continuing the application on social networks, Salvatore A. Catanese, Pasquale De Meo, and Emilio Ferrara [Salvatore, 2011] have applied web crawlers on Facebook and produced a source for later use of the community structure of Facebook. Web crawlers were also used to procure biological data from the web by a study of Ari Pirkola [Ari, 2007].

Furthermore, a number of researchers have put their efforts to improve and enhance the performance of web crawlers. Hybrid crawlers were suggested by Hetal J. Thankil et al. [Thankil, 2015] to improve the quality of service as it may overcome the limitation of the crawling algorithm. Different techniques to develop a crawler and to build an efficient crawler is detailed in a paper of Raja Iswary and Keshab Nath [Raja and Keshab, 2013], and an article of Trupti V. Udapure et al. [Trupti et al., 2014]. Dr. Jatinder Singh Bal et al. advised that choosing the right plan and building an adequate structure are more challenging than creating a capable web crawler. At the same time, efficient crawling algorithms should be deepened to provide better results and higher performance [Mini and Jatinder, 2014].

It can be seen that most of the mentioned studies have concentrated on using template-based crawling techniques; however, they reach a general limitation by this approach. Those techniques could not handle the layout diversity of objects from web pages. Therefore, in this paper, a web crawler model was proposed and implemented to extract article-like contents from 495 websites. It uses a machine learning approach with visual cues [Magallón-García et al., 2017], trivial HTML, and text-based features to filter out clutters. Given a web page Uniform Resource Locator (URL), the model can extract relevant titles and correspondent contents. The outcomes are promising for extracting article-like contents from websites, contributing to the fields of cybernetics, telecommunication, and physics.

3 System Architecture of Web Crawler

Template-based web crawlers are normally used to extract data from web-pages, however, specific template from each website is vary and challenging for this technique. Therefore, in this paper, a new web crawler is proposed to extract web-pages from blocks by blocks, attaching with each title and content, (as shown in Figure 1) making the crawling data produce contents similar to an article format.

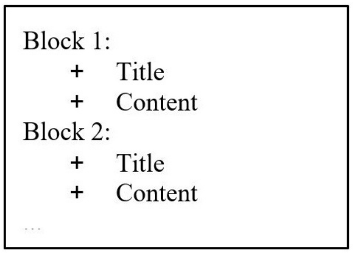

Block 1:
+ Title
+ Content
Block 2:
+ Title
+ Content
...

Figure 1. Data layout after web crawling.

The architecture of the topical web crawler designed and implemented in this paper is indicated in Figure 2.

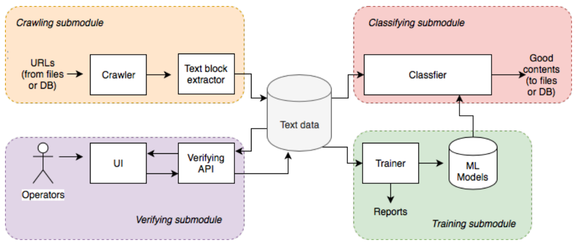


Figure 3. Proposed web crawler modules.

Figure 4. Website Sample.

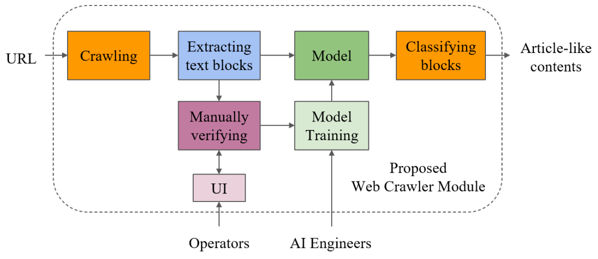


Figure 2. Proposed web crawler architecture.

The module is performed in a combination of tasks, including: Crawling text content from webpages; Extracting text blocks; Verifying manually; Training Machine Learning models; and Classifying blocks.

Its operation process is indicated in Figure 2 and described as follows.

1. Using URLs, websites were crawled in forms of HTML files, which includes different web pages;
2. Those pages were then extracted into a number of blocks, including pictures and texts; filter out and select only text blocks for the next experiment;
3. The text blocks were manually verified and labeled by operators, through a UI (user interface);
4. Only usable text blocks for crawling were inputted for a Machine Leering training model, implemented by AI Engineers;
5. From there, a model was built from the output of the Machine Learning process, which will be used (as a product) to classify blocks into article-like contents.

4 Proposed Web Crawler Module

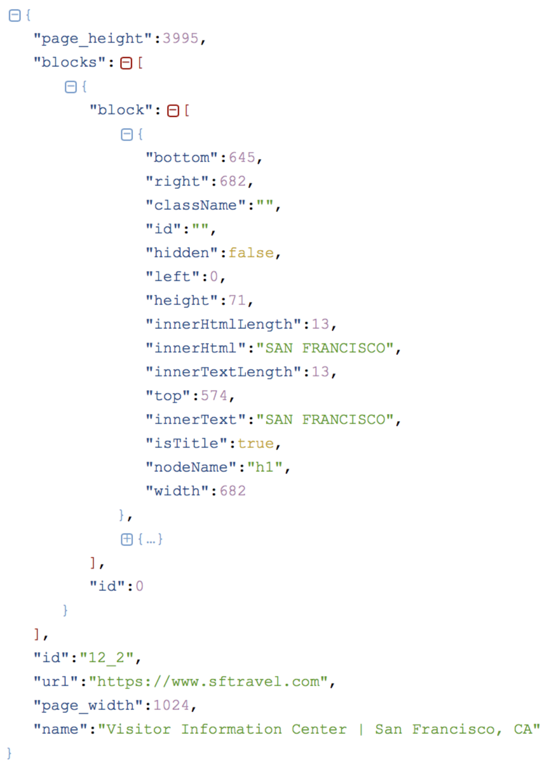
The proposed module of the entire crawler system is divided into 4 sub-modules, which are illustrated in Figure 3. Each sub-module implements specific tasks as detailed below.

1. Crawling: getting data from web pages, extracting text blocks, and saving them into a local database;
2. Verifying: verifying each text block by operators and submitting results into the database via an API;
3. Training: training Machine Learning models using data from database and export reports;
4. Classifying: classifying text blocks in the database using the trained Machine Learning model, exporting as files or database.

4.1 Crawling Sub-module

The crawling sub-module is designed to be implemented on the crawling server. By using URL, for example, a website sample is crawled as an HTML file, as shown in Figure. 4. The crawling sub-module runs as a batch in the background, taking input from CSV/JSON URL files or the database.

The output of this module is a JSON string that contains a web page's text blocks and other attributes, as indicated in Figure 5. Then, the string is inserted into a local database to verify it in the next step.



```
⊟{
   "page_height":3995,
   "blocks":⊟[
      ⊟{
         "block":⊟[
            ⊟{
               "bottom":645,
               "right":682,
               "className":"",
               "id":"",
               "hidden":false,
               "left":0,
               "height":71,
               "innerHtmlLength":13,
               "innerHtml":"SAN FRANCISCO",
               "innerTextLength":13,
               "top":574,
               "innerText":"SAN FRANCISCO",
               "isTitle":true,
               "nodeName":"h1",
               "width":682
            },
            ⊞{…}
         ],
         "id":0
      }
   ],
   "id":"12_2",
   "url":"https://www.sftravel.com",
   "page_width":1024,
   "name":"Visitor Information Center | San Francisco, CA"
}
```

Figure 5. Crawling Sub-module Output.

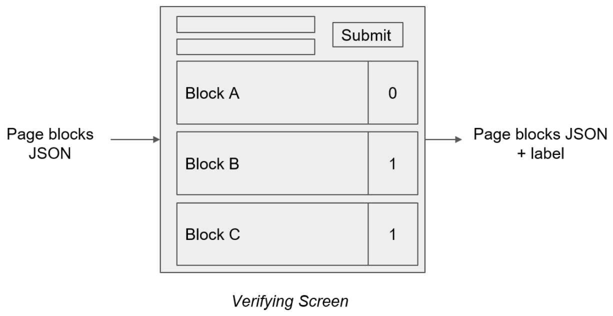


Figure 6. Verifying Sub-module.

4.2 Verifying Sub-module

The verifying sub-module is integrated into the admin console. This sub-module takes the input of JSON strings, which contain web pages' text blocks and their URLs. Its purpose is to insert a block label of the input and save it to the database, in order to provide data for a Machine Learning model, as presented in Figure 6.

4.3 Training Sub-module

4.3.1 Training Module Structure The training sub-module is implemented on an AI training server, which runs manually as a batch and monitored by AI engineers. This sub-module takes inputs of JSON blocks from the database, and outputs report logs and dumped models, as indicated in Figure 7.

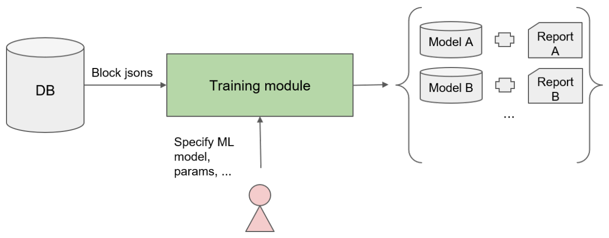


Figure 7. Training Sub-module.

4.3.2 Machine Learning Model: Support Vector Machine The Support Vector Machine (SVM) was used in this research to train the module to classify blocks. [Jankowski et al., 2009] As a supervised machine learning algorithm, SVM is commonly used for classification rather than regression purposes. [Kumon et al., 2007] It is widely used in various classification applications of input samples [Hieu and Hien, 2020a; Agarap, 2017; Hieu and Hien, 2020b], and become a state-of-the-art classifier. Let $\{(x_i, y_i)\}_{i=1}^{N}$ be a set of N training samples, where x_i is the ith sample in the input space x, and $y_i \in \{+1, -1\}$ is the class of x_i label. The SVM decision function which classifies a new test sample x is represented as

$$f(z) = \text{sgn}\left(\int_{i=1}^{N} \alpha_i y_i k(x_i, z) + b\right)$$

where z is presented as an unclassified sample, α_i is the Lagrange multiplier of a dual optimization problem which represents the separating hyperplane; $k(.\ ,.)$ indicates the kernel function which should satisfy the Mercer's condition; and b is the threshold parameter of hyperplane [Ribeiro, 2005].

The training sample x_i (with $\alpha_i > 0$) is called support vectors, and the SVM classifier finds the optimal hyperplane that maximizes the separating margin between two classes, as shown in Figure 8 [Song et al., 2002].

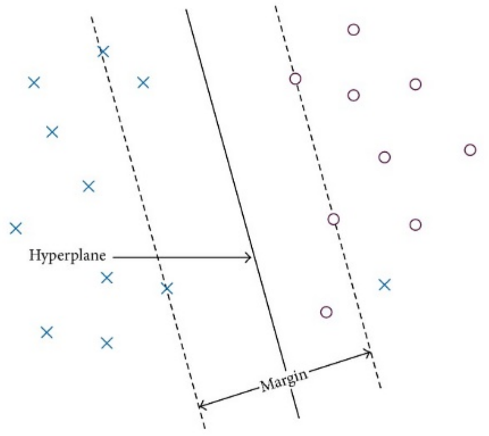


Figure 8. Optimal hyperplane of SVM in non-separable cases.

In this research, the fixed hyperparameter Gamma is 'scale'. For tuning, the Kernel used is RBF, sigmoid, linear, and poly, with a penalty from 0.01 to 1000, and label 0 weight from 1.0 to 2.5.

4.3.3 Tuning Method: Cross Validation During a machine learning process, the dataset is normally split into training and test sets. [Goras and Fira, 2009] Afterward, the training set is used to train the model and the test set is handled to evaluate the model performance. [Ge et al., 2017] However, this method may lead to variance problems, in which the accuracy obtained on one test is somewhat different from the accuracy obtained on another test set using the same algorithm. [Sebastian, 2018]

In order to solve this issue, K-Fold Cross-Validation is used in this research to evaluate the performance. From there, the data is divided into K folds, and K-1 sets are used for training whereas the remaining set is used for testing. The algorithm is consequently trained and tested K times, each time using a new set as the test set, and then using the remaining sets for training. Eventually, the K-Fold Cross-Validation result is the average of the results collected from each set. [Sylvain and Alain, 2009]

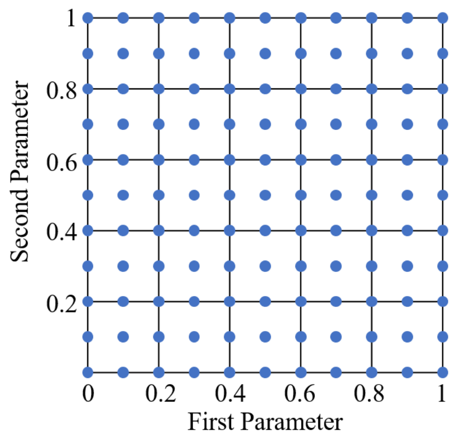


Figure 10. Grid Search space illustration.

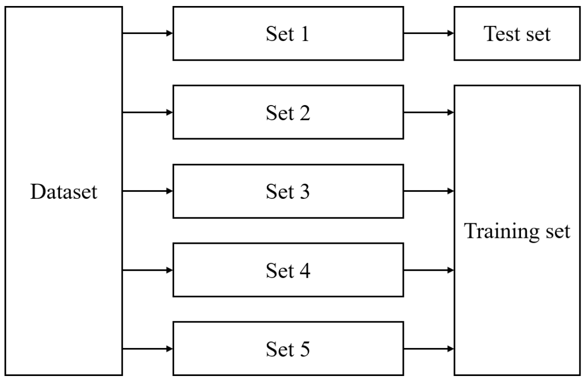


Figure 9. Cross-validation.

For example, Figure 9 illustrates 5-fold cross-validation, where the data is split into 5 sets, from Set 1 to Set 5. The algorithm is then trained and tested 5 times. In the first fold, Set 1 is used as a testing set, and the remaining sets are used as training sets. In the second fold, for example, while Set 2 is used for testing, the rest of the Sets are used for training. The process continues continuously until every set is used for testing at least once. The final result is conducted by the average of results taken by all folds. The cross-validation method not only can get rid of the variance but also can find the variance in the overall result by using the standard deviation of the results.

4.3.4 Parameter Selection: Grid Search A machine learning model normally has two types of parameters, one is learned through the process, and the other one is the hyperparameter that we pass to the model. The value of these hyperparameters is normally set in random to define what parameters result in the best performance. [Melnikov and Barabanov, 2016] However, this approach reaches some common limits because one algorithm can perform better or worse than the other, depending on different sets of parameters, leading to an inadequate comparison. Therefore, Grid Search is an algorithm invented to automatically defines the best parameters for each particular model. [Petro and Pavlo, 2019]

Grid search is a process of scanning data to configure the best parameters for a given model. Depending on the model type used, certain parameters are required. Therefore, grid search not only can apply to one model type but also can apply across machine learning to calculate the optimum parameters. Grid search then builds a model based on all possible parameter combinations. It traverses each parameter combination and stores a model for each combination. [Petro and Pavlo, 2019]

In the settings for Grid Search in this research, the optimized score F-beta and beta are 0.5 (prioritizing Precision), with the number of folds is 5. There are two main steps in this process: Selecting kernel, penalty, and class weight; and the Fine-tuning penalty.

4.4 Classifying Sub-module

This sub-module classifies text blocks in the database, using a trained Machine Learning model. Then, it exports outputs in the forms of files or databases, as presented in Figure 11. At the end of the process, the receive results are article-like contents, which meets the requirement to handle the layout diversity of objects from web pages.

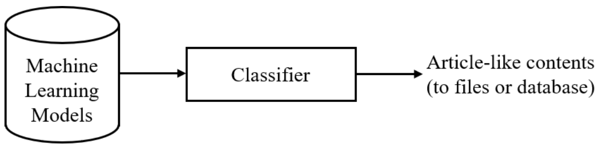


Figure 11. Classifying Sub-module.

5 Experiment Results

5.1 Dataset Collection

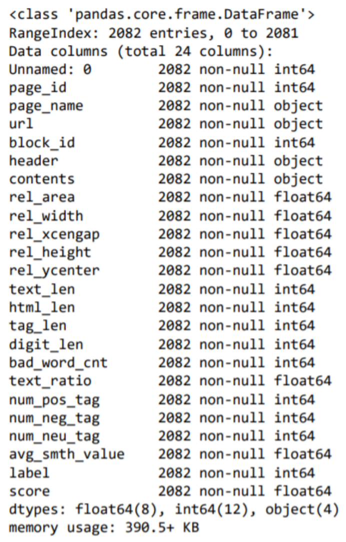
The website dataset was collected from 495 URLs, which is corresponding to 495 web pages. The final blocks extracted from the Crawling Sub-module are 2082 text blocks, with 777 blocks are classified as "good" (accounting for 37,32%, containing content information of the website) and 1305 blocks are classified as "bad" (accounting for 62.68%, containing navigations, contact boxes, Ads, etc.). There are also 15 numeric features and 1 compound score. The dataset feature details are shown in Figure 12.

5.2 Experiments

To standardize values, the dataset which contains raw HTML files was preprocessed by using Selenium [Shaumik and Pradeep, 2019]. The next step was to extract the files into length and width values, before clustering by Density-based spatial clustering of applications with noise (DBSCAN) algorithm [Ester, 1996]. Then,

Table 2. Training and testing result of the experimented model

	Accuracy (%)	Precision (%)	Recall (%)
Training	86.67	89.08	76.76
Testing	83.93	90.02	72.19



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2082 entries, 0 to 2081
Data columns (total 24 columns):
Unnamed: 0        2082 non-null int64
page_id           2082 non-null int64
page_name         2082 non-null object
url               2082 non-null object
block_id          2082 non-null int64
header            2082 non-null object
contents          2082 non-null object
rel_area          2082 non-null float64
rel_width         2082 non-null float64
rel_xcengap       2082 non-null float64
rel_height        2082 non-null float64
rel_ycenter       2082 non-null float64
text_len          2082 non-null int64
html_len          2082 non-null int64
tag_len           2082 non-null int64
digit_len         2082 non-null int64
bad_word_cnt      2082 non-null int64
text_ratio        2082 non-null float64
num_pos_tag       2082 non-null int64
num_neg_tag       2082 non-null int64
num_neu_tag       2082 non-null int64
avg_smth_value    2082 non-null float64
label             2082 non-null int64
score             2082 non-null float64
dtypes: float64(8), int64(12), object(4)
memory usage: 390.5+ KB
```

Figure 12. Dataset details.

the blocks were labeled and extracted their features. Finally, the dataset was split into an unstratified training test with 20% samples and was delivered to the training process. Therefore, there is a total of 1665 training text blocks and 417 testing text blocks in the end.

First, the model suggested by GridSearchCV was used to train and test the dataset. The features used include Kernel SVM, gamma = 'scale', class_weight= {0: 1.9}, and penalty = 1000. Using this model, the training set obtained remarkably high accuracy and precision scores. However, there were 3 noticeable drops (with over 10%) on the testing set. The result of this model is presented in Table 1.

Table 1. Training and testing result of the model suggested from GridSearchCV

	Accuracy (%)	Precision (%)	Recall (%)
Training	92.01	96.43	83.53
Testing	81.06	84.62	70.59

Then, the authors experimented another model with different features, with Kernel SVM, gamma = 'scale', class_weight= {0: 1.5}, and penalty = 10. By using this model, the accuracy, precision, and recall were lower than the former model. However, it solves the overfitting issue that happened in the model suggested by GridSearchCV (less than 5%). The result of this model is indicated in Table 2.

Comparing the results of 2 models, the GirdSearchCV is more optimal for crawling websites to create article-like contents. To indicate the result comprehensively, the system was trained on this model using different training set sizes, 100, 500, 1000, and 1665 samples. A comparative figure is presented for the accuracy, precision, and recall of training and testing scores in Table 3 and Figure 13.

Table 3. Comparative training and testing result on different training set sizes

Training set size	**Result**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
100	Training	99.23	99.40	99.11
	Testing	68.59	65.22	64.17
500	Training	93.81	99.44	85.51
	Testing	79.38	80.24	71.66
1000	Training	93.32	97.77	85.61
	Testing	81.29	83.85	72.19
1665	Training	92.01	96.43	83.53
	Testing	81.06	84.62	70.59

It can be seen from the figure that the bigger the training set size, the less efficient on the training set, but the more effective on the testing set, regarding the accuracy, precision, and recall values.

6 Conclusions

In this paper, a web crawler model was designed and developed, which extracted websites into article-like contents. Firstly, the web crawler module was designed into 4 different sub-modules, crawling, verifying, training, and classifying. The website dataset was collected from 495 URLs, then extracted into 2082 text blocks, and split its 20 percent for the training set. The model used the Support Vector Machine model to train, Cross-Validation for tuning, and Grid Search for parameter selection. After experience on different models, the optimal suggested is the GridSearchCV, where it may deliver 92.01% for accuracy and 96.43% for precision on the training set, and 81.06%, 84.62% respectively on the testing set. It is also noticeable that the efficiency, regarding the accuracy, precision, and recall values are higher when the training size increases. The system can accommodate to reduce the workload of management and optimization in search engine systems, contributing to the development of cybernetics, telecommunication,

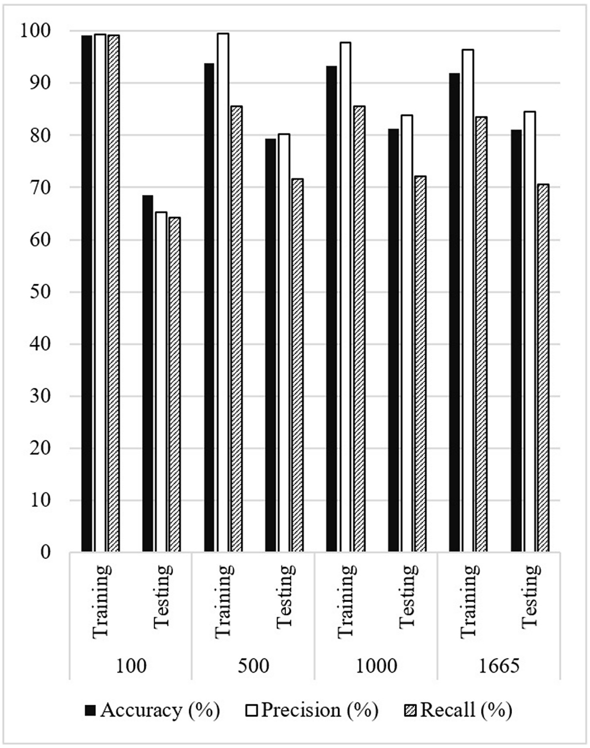


Figure 13. Comparative chart on training and testing result of different training set sizes.

and physics. The outcomes of this study open up new avenues for future research and can serve as a hypothetical source for future web crawler systems to extract article-like contents.

7 Future Scope

Although substantial efforts have been made in the past [Pramudita, 2020; Kapil and Mukesh, 2019; Petro and Pavlo, 2019], our research proposed high-efficiency models for web crawler to extract article-like contents. In future research, attempt gears towards applying more machine learning methods, including Ensemble and Neural Net, to improve the performance of the model. In order to reach a higher accuracy, a sitemap scan to obtain main images is needed to upgrade the current work. Future work will also focus on classifying blocks into 'header', 'menu', 'footer', 'ads', 'left-bar', etc. and extracting all page attributes (keyword, description, breadcrumb, etc.) to investigate a higher performance of web crawler model to extract article-like contents.

References

Agarap, A. F. (2017). An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification. *arXiv*.

Ari, P. (2007). Focused crawling: A means to acquire biological data from the web. *University of Tampere Finland*, **978**.

Avinash, N. B., Harsha, A. B., and Meshram, B. B. (2010). Intelligent web agent for search engines. *International Conference on Trends and Advances in Computation and Engineering*.

BeenaMahar and Jha, C. K. (2015). A comparative study on web crawling for searching hidden web. *International Journal of Computer Science and Information Technologies*, **6** (3), pp. 2159–2163.

Bruce, C., Donald, M., and Trevor, S. (2009). Search engines information retrieval in practice.

Christopher, O. and Marc, N. (2010). Web crawling. *now the essence of knowledge*, **4** (3), pp. 2010.

Desai, K. (2017). Web crawler: Review of different types of web crawler, its issues, applications and research opportunities. *International Journal of Advanced Research in Computer Science*, **8**, pp. 1199–1202.

Dhiraj, K. and Satish, K. (2012). Web crawler: A review. *International Journal of Computer Science and Management Studies*, **12**.

Ester, M. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. pp. 226–231. Second International Conference on Knowledge Discovery and Data Mining (KDD-96).

Ge, Z., Song, Z., Ding, S. X., and Huang, B. (2017). Data mining and analytics in the process industry: The role of machine learning. *IEEE Access*, **5**, pp. 20590–20616.

Goras, L. and Fira, C. M. (2009). Preprocessing method for improving ecg signal classification and compression validation. *4th International Conference on Physics and Control*.

Hieu, N. V. and Hien, N. L. H. (2020a). Automatic plant image identification of vietnamese species using deep learning models. *International Journal of Engineering Trends and Technology*, **68** (4), pp. 25–31.

Hieu, N. V. and Hien, N. L. H. (2020b). Recognition of plant species using deep convolutional feature extraction. *International Journal on Emerging Technologies*, **11** (3), pp. 904–910.

Jain, S. S. and Subodh, P. G. (2018). A methodical study of web crawler. *International Journal of Engineering Research and Applications*, **8** (11), pp. 1–8.

Jankowski, S., Currenti, G., Napoli, R., Szymanski1, Z., Fortuna, L., and Negro, C. D. (2009). Modelling volcanomagnetic dynamics by recurrent least-squares support vector machines. *4th International Conference on Physics and Control*.

Kapil and Mukesh, Y. (2019). Design of a novel interface for a web crawler. *International Journal of Electronics Engineering*, **11** (1), pp. 952–958.

Kausar, M. A., Dhaka, V. S., and Sanjeev, K. S. (2013). Web crawler: A review. *International Journal of Computer Applications*, **63** (2).

Kobayashi, M. and Takeda, K. (2000). Information retrieval on the web. *ACM Computing Surveys (ACM Press)*, **32** (2), pp. 144–173.

Komal and Ashutosh, D. (2016). Design issues in web crawlers and review of parallel crawlers. *International Journal of Science and Research*, **5** (6), pp. 61–64.

Kumon, M., Ito, Y., Nakashima, T., Shimoda, T., and Ishitobi, M. (2007). Sound source classification using support vector machine. *9th IFAC Workshop Adaptation and Learning in Control and Signal Processing*.

Magallón-García, D. A., Jaimes-Reátegui, R., Huerta-Cuellar, G., Gallegos-Infante, L. A., Soria-Fregoso, C., and García-López, J. H. (2017). Study of multistable visual perception using the synergetic model. *Cybernetics and Physics*, **6** (3).

Martin, E., Hans-Peterand, K., and Matthias, S. (2004). Accurate and efficient crawling for relevant websites. *30th VLDB Conference Toronto, Canada*, pp. 396–407.

Melnikov, A. and Barabanov, A. (2016). Guaranteed estimation of speech fundamental frequency with bounded complexity algorithm. *Cybernetics and Physics*, **5** (1).

Minas, G. (2010). *Measurement of Online Social Networks*. University of California, Irvine.

Mini, S. A. and Jatinder, S. B. (2014). Web crawler: Extracting the web data. *International Journal of Computer Trends and Technology*, **13** (3).

Mridul, B. S. and Bharne, S. (2016). A survey on various kinds of web crawlers and intelligent crawler. *International Journal of Scientific Engineering and Applied Science*, **2** (3).

Pant, G., Srinivasan, P., and Menczer, F. (2004). Crawling the web. In *Web Dynamics*. Springer, Berlin, Heidelberg.

Petro, L. and Pavlo, L. (2019). Grid search, random search, genetic algorithm: A big comparison for nas.

Pinkerton, B. (1994). *Finding What People Want: Experiences with the WebCrawler*. Second International World Wide Web Conference, Chicago, Illinois, USA.

Pramudita, Y. D. (2020). Extraction system web content sports new based on web crawler multi thread. *Journal of Physics: Conference Series*, **1569**.

Priyanka, S. (2019). Mercator as a web crawler. International Journal of Computer Science Issues.

Raja, I. and Keshab, N. (2013). Web crawler. *International Journal of Advanced Research in Computer and Communication Engineering*, **2** (10).

Ribeiro, B. (2005). Support vector machines for quality monitoring in a plastic injection molding process. *IEEE Transactions on Systems, Man and Cybernetics C.*, pp. 401–410.

Salvatore, A. C. (2011). Crawling facebook for social network analysis purposes. *WIMS*, **11** (25-27).

Sebastian, R. (2018). *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning*. University of Wisconsin–Madison.

Shaumik, D. and Pradeep, K. (2019). Getting started with web automation testing using selenium and python.

Song, Q., Hu, W., and Xie, W. (2002). Robust support vector machine with bullet hole image classification. *IEEE Transactions on Systems, Man and Cybernetics C.*, pp. 440–448.

Sylvain, A. and Alain, C. (2009). A survey of cross-validation procedures for model selection.

Thankil, J. H. (2015). Domain-specific web crawler: A survey. *International Journal of Innovative Research in Science, Engineering and Technology*, **4**.

Trupti, V. U., Ravindra, D. K., and Rajesh, C. D. (2014). Study of web crawler and its different types. *IOSR Journal of Computer Engineering (IOSR-JCE)*, **16** (1), pp. 1–5.

Vladislav, S. and Torsten, S. (2001). Design and implementation of a high-performance distributed web crawler. *NSF CAREER Award*, **CCR-0093400**.