NBCE: A Neo4j-Based Content Extraction Algorithm in Threat Intelligence Web Pages

Xiaoyang Li[1,a], Mengming Li[1,b], Rongfeng Zheng[2,c], Anmin Zhou[1,d] and Liang Liu[1,e,*]

[1]College of Cybersecurity, Sichuan University, Chengdu, Sichuan, China
[2]College of Electronics and Information Engineering, Sichuan University, Chengdu, Sichuan, China
a. shawnlee97@163.com, b. limengmingx@sina.cn, c.qswhs@ foxmail.com
d. zhouanmin@scu.edu.cn, e. liangzhai118@163.com
**corresponding author: Liang Liu*

Keywords: Main content extraction, threat intelligence, Neo4j, machine learning.

Abstract: Main content extraction is a widely used technique in web crawler, search engines and so on to extract the main content of web pages as well as discarding other complementary and decorative components. By extracting the main content, irrelevant and redundant information can be ignored hence reducing the complexity of data processing and improving the efficiency of further analysis. Among the existing methods tackling this problem, solutions are designed to satisfy the different requirements of various groups. For instance, companies specialized in content extraction always focus more on efficiency and accuracy while others may concentrate more on practicality. In our proposed method, we innovatively present a neo4j-based content extraction algorithm (NBCE) in threat intelligence websites. The NBCE algorithm initially transforms the HTML source code into the form of the tree structure. Then the triples extracted from the HTML tree are used to construct a graph based on neo4j database. Finally, by deciding whether a node is the main content node or not, the main content of the given web page can be extracted. The availability of the proposed method is validated through a set of experiments conducted on a threat-intelligence-related database.

1. Introduction

With the increasing complexity of World Wide Web, main content extraction, as an effective way to select valuable information from web pages while ignoring other non-informative elements, deserves great attention. However, the detailed requirements of web page content extraction vary from one group to another. On the one hand, well-organized corporations with abundant human resources are able to obtain large amounts of samples by advanced and accurate algorithms. On the other hand, people in small businesses always concentrate more on specific applications and therefore require task-oriented algorithms. For example, in cybersecurity, the extracted main content can be used to

collect and analyze low-level IOCs[7, 8] and high-level IOCs [11] utilized by hackers. Therefore, main content extraction has turned out to be topical issues in cybersecurity.

What's more, in the applications of general web crawler, a great of time is spent on parsing the HTML structure. However, for other web crawlers aimed at particular fields, the workload is much lower and they only have to be operated several times to satisfy the specific requirements. The reason is that for those small-size enterprises, they don't engage in web page analyzing but depend on main content extracted from web pages to support follow-up operations. Therefore, more attention will be focused on accuracy and practicality than processing time. Moreover, it is difficult for them to design a context extraction algorithm based on annotated datasets on account of their limited human resources and budgets. Concerning this, a reliable algorithm with simple rules and high robustness is still sought after.

In this paper, we propose a simple but reliable neo4j-based content extraction algorithm (NBCE) for the accurate and efficient extraction of cybersecurity-related web content. Our method consists of three parts. In the first step, the HTML source code is transformed into a simple graph using neo4j database according to its main structure. Then, specific rules are set to discard null nodes to reduce redundancy. Finally, we approach the issue of content extraction as a binary classification question, using machine learning on the set of nodes obtained from former steps to extract the unique main content node, and thereby extracting the main content of the given web page. Moreover, we perform 10-folds cross-validation on a set of experiments to evaluate our proposed method.

The rest of the paper is organized as follows. Section II reviews the related works tackling the task of main content extraction in web pages. In section III, the proposed NBCE algorithm is explained in detail. The experiments to evaluate the performance of the novel method are described in section IV. A brief conclusion and future explorations are presented in section V.

2. Related Work

Main content extraction, as an indispensable technique in many applications such as web content aggregation, web crawler and search engine, has attracted the attention of numerous researchers. There are many approaches aiming at this problem [5, 6, 14]. Kreuzer et al. [10] concluded a taxonomy of some existing methods according to the type of information used for extraction: textual information, visual information, DOM Tree and so on.

The main idea of regarding textual information as the chief gauge for content extraction is that the text density of main content in a web page is much higher than other blocks. Weninger et al. [13] put forward the Content Extraction via Tag Ratios (CETR), in which the ratio of characters and labels inside each label is calculated to serve as a determinant. Besides, in [3] a quantitative linguistic approach is employed with consideration of other HTML metrics such as link density.

Visual methods such as VIPS [4] and its derivatives [12] make use of the visual tree structure of HTML to extract the main content. By dividing the web page into blocks, the ratio of text nodes and leaf nodes in each block is calculated to decide whether it is the main content or not. In addition, Burget et al. [9] combined CSS feature based on the VIPS algorithm, which improves the performance to a certain degree.

Bar-Yossef et al. [2] regarded the template in web pag es as noise. They detected the template by analyzing the DOM tree and further extracted the main content. Many other methods based on the DOM tree [1, 15] have also turned out to be effective.

Although many studies have been conducted, few of them take the relation between HTML code and graph into consideration as our proposed NBCE algorithm.

3. Method

We propose a Neo4j-Based Content Extraction algorithm (NBCE) to extract the main content from the given threat intelligence web page. The proposed NBCE approach can be divided into HTML-Neo4j transformation phase, compression phase and content extraction phase, as shown in Figure 1. To transform HTML source code into a graph, we take advantage of the specific tree structure and extract the triples necessary for constructing a graph in neo4j database. Based on the constructed graph, we make further improvements by performing node-level and branch-level compression to reduce redundancy. Finally, the processed nodes can serve as the input of machine learning to train models and then extract the expected main content. The following subsections will describe these steps in detail.

3.1. HTML-Neo4j Transformation

HTML always presents predominantly unstructured data with structured tags. Triples necessary for constructing a graph can be extracted from related tags according to their hierarchical relationships. Algorithm 1 and Algorithm 2 represent the details in the transformation phase. In algorithm 1, all the nodes in the HTML tree are traversed to find the relation between two directedly connected nodes. For the first-level tags under the original HTML source code, these tags and their corresponding sequences are recorded for further employment. Then, similarly to the previous step, the subtags under first-level tags and their sequences are also captured. Therefore, through traversing, the relations between all nodes can be represented in the form of [sub_node, "sub_tag",[children, number]], where “sub_tag” represents the relation between two directly connected nodes and “number” represents the sequence of sub nodes. Furthermore, in algorithm 2, all the triple nodes in the HTML can be aggregated. Hence the HTML source code can be transformed into the form of a tree structure. Figure 2 displays a transformation sample in detail. Finally, RDF triples are extracted and stored in neo4j graph database to construct the graph of the given web page. The standard template of RDF triple used in neo4j database is described in Figure 3, where “src” and “dst” represent different nodes in a graph while “r” represents the relation between two nodes as well as an edge in a graph.

3.2. Node-level and Branch-level Compression

There exist two kinds of nodes connected directly to a null node in the generated graph, namely end nodes and branch nodes. As we can see from Figure 4, node B without any child node is connected to a null node and the null node in turn is connected to another null node. Therefore, node B can be regarded as an end node. Whereas in Figure 5, node C is connected to a series of leaf nodes at one end while connected to a null node at another. Hence it is defined as a branch node. On account of the numerous null nodes brought about by HTML tags, we perform node-level and branch-level compression in our proposed approach. The main idea of compression is discarding the unnecessary null node to reduce redundancy. More specifically, in node-level compression, the null node connected directly to an end node is discarded and the end note is then connected to the ancestor of the null node. Similarly, in branch-level compression, the null node connected directly to the branch node is also removed and its ancestor is connected to the branch node. The detailed procedures can be seen in Algorithm 3.

By removing these redundant nodes, we eliminate the influence of data discretization brought about by HTML tags. Furthermore, because of the particular structure of HTML, the nodes related to similar topics are always connected to the same or adjacent nodes. Therefore, those nodes that

contain main content must be connected to the same node directly or within a certain distance, which is the underlying principle of our approach.

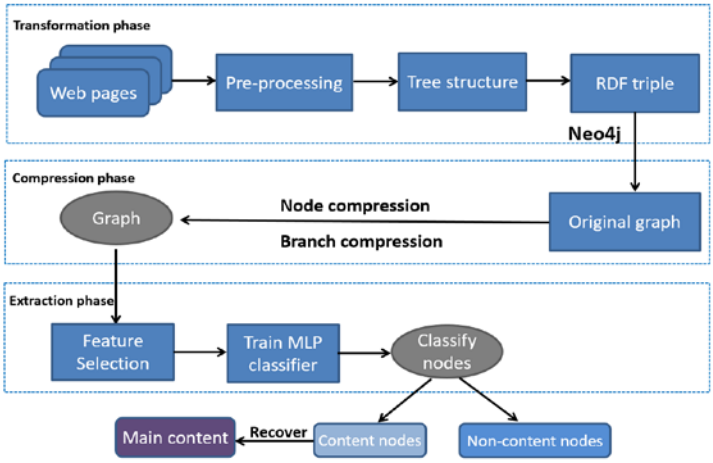

Figure 1: Processing phases for main content extraction.

```
<html>                                  [HTML_TOTAL, 'sub_tag', [SUB_0, 0]],
  <div>                                 [HTML_TOTAL, 'sub_tag', [<p>  IoCs: </p>, 1]],
    <p> here is the reason :</p>        [HTML_TOTAL, 'sub_tag', [SUB_2, 2]],
    <p> 1. we love peace</p>
    <p> 2. we hate war</p>
  </div>                                [SUB_0, 'sub_tag', [<p>here is the reason: </p>, 0]],
  <p>IoCs:</p>                          [SUB_0, 'sub_tag', [<p>  1. we love peace  </p>, 1]],
  <div>                                 [SUB_0, 'sub_tag', [<p>  2. we hate war    </p>, 2]],
    <p> domain-1 </p>
    <p> ip-2 </p>
  </div>                                [SUB_1, 'sub_tag', [<p>   domain-1 </p>, 0]],
</html>                                 [SUB_2, 'sub_tag', [<p>   ip-2    </p>, 1]]]
```

Figure 2: A sample of transforming HTML into the tree structure.

```
result = {
        "src": {
            "html_tag": src_main_tag,
            "unique_id": src_unique_id
        },
        'r': {
            'name': "sub_tag"
        },
        'dst': {
            "html_tag": dst_main_tag,
            "context": dst_raw_context,
            "unique_id": dst_unique_id,
            "children_sequence": children_sequence
        }
    }
```

Figure 3: The standard template of RDF triple.

3.3.Main Content Extraction

After the compression phase, redundant null nodes are discarded. Generally, in a graph, large amounts of nodes that contain main content may connect to a null node directly or within a certain distance, and therefore, this node is defined as the main content node. More specifically, there is only one content node in a web page. Besides, the content node extracted from a web page is corresponding to all the tags containing the main content in HTML structure. Therefore, according to the sequence recorded in the former steps, the main content can be recovered from the nodes connected to the content node.

In this way, the problem of the main content extraction is then transformed into the problem of main content node extraction. In our approach, we consider the issue as a binary classification question hence the nodes can be classified into content nodes and non-content nodes. Traditionally, this kind of binary classification questions can be solved by machine learning. Consequently in our approach, we extract several distinct features between content and non-content nodes to perform machine learning. The features we selected are based on the following rules to figure out whether the null node is content node or not:

- The number of nodes connected directly to the null node.
- The average text length of the nodes connected directly to the null node.

```
  Algorithm 1. Elstablish_triple_nodes
  Input: HTML Source Code
  Output: the tree structure of the given HTML
  1 var_triple_node ← first-level nodes
  1 for sub_node in var_triple_node do
  2     if not_hasChild(sub_node) then
  3         continue
  4     else
  5         children_result = get_children_tags(sub_node)
  6         number = 0
  7         var_all_triple.append([sub_node,
"sub_tag",[children, number]])
  8         number = number+1
  9         var_all_triple =
elstablish_trip_nodes(children_result, var_all_triple)
  10    end
  11 end

  Algorithm 2. Build_tree_structure
  Input: HTML Source Code
  Output: the tree structure of the given HTML
  1 var_all_triple = []
  2 var_triple_node = get_children_tags(html)
  3 soup = BeautifulSoup(html)
  4 for each triple_node in var_triple_node do
  5     var_all_triple.append([soup, "sub_tag",
[triple_node, number]])
  6     number = number+1
  7 end
  8 var_all_triple =
elstablish_triple_nodes(var_triple_node, var_all_triple)
```

```
Algorithm 3. End/branch compression
Input: all the nodes
Output: compressed nodes
1 for each node in end/branch node do
2     if node→parent = null then
3         node→parent = node→parent→parent
4     end
5 end
```

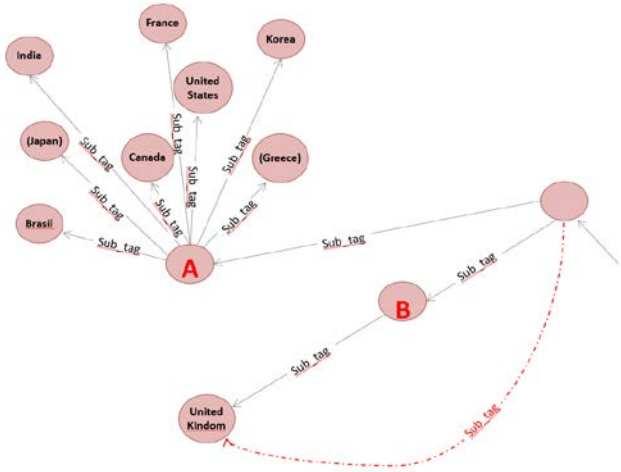

Figure 4: An example of end node.

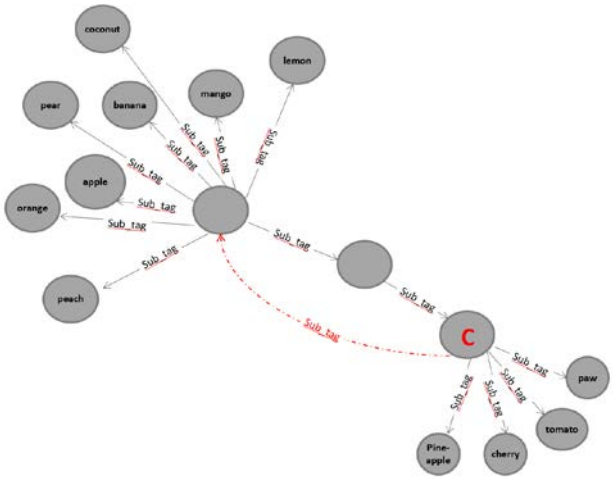

Figure 5: An example of branch node.

On the one hand, the main content node always corresponds to more nodes due to the particular structure of a website. On the other hand, some websites contain recommendation parts, which can also be transformed into graphs. However, as there is only one main content node for each web page, this kind of part containing less information with shorter text length should not be regarded as the main content.

Additionally, in a graph transformed by a website, the number of content nodes is far less than that of non-content nodes. Therefore, in our proposed NBCE algorithm, we depend on the MLP

model to perform classification, which will be explained in detail in Section IV. And from the main content nodes extracted from machine learning, the main content of the web page can be recovered by using the sequences recorded in former steps.

4. Experiment

To evaluate the availability of our proposed NBCE algorithm, two experiments are conducted on the same cybersecurity-related dataset. The first one confirms the superiority of our proposed algorithm while another is performed to select the most suitable machine learning model by applying extensively used measures such as precision, recall and F1-score.

4.1. Dataset

To evaluate the effectiveness of our proposed method, we collected 100 web pages from 7 authoritative threat intelligence publishing platforms as the dataset. Each sample contains the source content (HTML) of the web pages, which is then pre-processed to feed the need of the neo4j database and finally the HTML is transformed into a graph. All the nodes that match the criterion described in section III. are labeled as the main content nodes, the rest as non-content nodes. To perform the 10-fold cross-validation, we then divide the dataset into 4 parts, in which 75% are for training and the remaining for testing. In addition, traditional measures such as precision, recall, F1-score and processing time are used to evaluate the performance of the classifiers.

4.2. Experimental Design

To further illustrate the advantages of our proposed method NBCE, another extensively used method VIPS described in [4] is performed on the same dataset for comparison. Different from our method, this method is based on vision and text leafs ratio. In VIPS method, the web page is initially divided into several blocks according to its visual content. Then in each block, the ratio of text nodes and leaf nodes is calculated to decide whether the given block is the main content or not. And the algorithm is regarded as effective only when the extracted content covers 95% of the manually selected main content. The comparison is conducted in terms of precision and processing time.

Another experiment is designed to evaluate the superiority of the chosen MLP classifier. By comparing with some other classifiers such as SVM and Naïve Bayes in terms of precision, recall and F1-score, the most suitable one is chosen.

4.3. Experimental Result

We use three-quarters of the dataset as the training set and the rest as the testing set to perform 10-fold cross-validation. The practiced model is then used to extract the main content of a given web page. Table 1 displays the extracting result of our proposed method and vision-based method VIPS [4]. The result includes the precision rate and the average cost of time for each web page.

As we can see from the experimental results, our proposed NBCE method has a better performance in terms of precision. Although it's at the cost of processing time, the precision of our method is 7% higher than the old one.

Table 1: The result of the proposed algorithm and VIPS.

	Precision	Cost of time
NBCE	93%	617ms
VIPS	86%	425ms

Table 2: The performance of different classifiers.

Models	Precision	Recall	F1-score
MLP	92%	78%	84%
SVM	70%	84%	76%
Naïve Bayes	50%	32%	39%

However, in task-oriented corporations, precision is of more importance than processing time. In our method, the precision is up to 93%, 7% higher than the old one whereas the delay is no more than 200ms. Therefore, this degree of delay is comparatively acceptable.

As to the performance of different classification models. Table 2 enlightens the gap between three classifiers in the given scenario. As we can see from the results, the precision of MLP is up to 92%, outperforming SVM and Naïve Bayes. Besides, the F1-score of MLP is also 8% and 45% higher than SVM and Naïve Bayes respectively. Therefore, we can conclude that MLP is the most suitable one among the three classifiers to extract the main content of a web page in the given scenario. The possible reason is that MLP is characterized by its fast convergence in small-size datasets. Additionally, we make an assumption that the limitation of our dataset may also have an influence. That is to say, when applied in larger datasets, other models may have better performance.

5. Conclusions

In this paper, we have proposed a method named NBCE to extract main content from threat intelligence web pages based on graph database neo4j. It makes full use of the tree structure of HTML and transforms the HTML source code into a graph based on neo4j database. Regarding the problem of main content extraction as a binary classification problem, several features are combined to decide whether a node is a content node or not. A set of experiments show that our method achieves competitive results on threat intelligence dataset, reaching 93% in precision. The results also reflect that in terms of classification models, the MLP model outperforms other traditional models in the main content extraction task. With the high precision in threat intelligence, we suggest that when applied in other particular websites such as pharmaceuticals sites, our NBCE algorithm will also have good performance. Future work could be done on expanding the dataset as well as improving its performance in terms of efficiency.

References

[1] Alarte, J., Insa, D., Silva, J., Tamarit, S.: Site-level web template extraction based on DOM analysis. In: Mazzara, M., Voronkov, A. (eds.) PSI 2015. LNCS, vol. 9609, pp. 36–49. Springer, Cham (2016).

[2] Bar-Yossef, Z., Rajagopalan, S.: Template detection via data mining and its appli- cations. In: Proceedings of the 11th International Conference on World Wide Web (WWW 2002), pp. 580–591. ACM, New York (2002).

[3] C. Kohlsch¨utter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in Proceedings ofthe third ACM international conference on Web search and data mining - WSDM '10, 2010, p. 441.

[4] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, "Extracting Content Structure for Web Pages Based on Visual Representation," Proceedings of the 5th Asia-Pacific web conference on Web technologies and applications, pp. 406–417, 2003.

[5] Ferraresi, A., Zanchetta, E., Baroni, M., Bernardini, S.: Introducing and evaluating ukWaC, a very large web-derived corpus of English. In: Proceedings of the 4th Web as Corpus Workshop (WAC-4), pp. 47–54 (2008).

[6] Insa, D., Silva, J., Tamarit, S.: Using the words/leafs ratio in the DOM tree for content extraction. J. Log. Algebr. Program. 82(8), 311–325 (2013).

[7] Li, K., Wen, H., Li, H., Zhu, H. and Sun, Limin. (2018). Security OSIF: Toward Automatic Discovery and Analysis of Event Based Cyber Threat Intelligence. 741-747. 10.1109/SmartWorld.2018.00142.

[8] Liao, X., Yuan, K., Wang, X. F., Li, Z., & Beyah, R.. (2016). Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence. the 2016 ACM SIGSAC Conference. ACM.

[9] R. Burget and I. Rudolfova, "Web Page Element Classification Based on Visual Features," 2009 First Asian Conference on Intelligent Information and Database Systems, pp. 67–72, 2009.

[10] R. Kreuzer, J. Hage, and A. Feelders, "A quantitative comparison of semantic web page segmentation approaches," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9114, 2015, pp. 374–391.

[11] Umara Noor, Zahid Anwar, Asad Waqar Malik, Sharifullah Khan, Shahzad Saleem, "A machine learning framework for investigating data breaches based on semantic analysis of adversary's attack patterns in threat intelligence repositories," Future Generation Computer Systems, vol. 95, pp. 467-487, 2019.

[12] W. Liu, X. Meng, and W. Meng, "ViDE: A vision-based approach for deep web data extraction," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 3, pp. 447–460, 2010.

[13] Weninger, T., Henry Hsu, W., Han, J.: CETR: Content Extraction via Tag Ratios. In: Rappa, M., Jones, P., Freire, J., Chakrabarti, S. (eds.) Proceedings of the 19th International Conference on World Wide Web (WWW 2010), pp. 971–980. ACM, April 2010.

[14] Wu, S., Liu, J., Fan, J.: Automatic web content extraction by combination of learning and grouping. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, pp. 1264–1274. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2015).

[15] Yi, L., Liu, B., Li, X.: Eliminating noisy information in web pages for data mining. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD 2003), pp. 296–305. ACM, New York (2003).