Locating Product Information from the Web using Simhash Fingerprints[i]

Tuan-Anh N. Pham[1], Khanh-Van Nguyen[2]
[1]School of Computer Engineering, Nanyang Technological University, Singapore
[2]School of Information and Communication Technology, Hanoi University of Science and Technology
Email: pham0070@e.ntu.edu.sg, vannk@soict.hust.edu.vn

***Abstract* -We considerthe problem of creating efficient search schemes that are specialized for product information; this is a very important issue given the explosive growth of commercial websites and Internet-based services. We share the observation in PEWeb [24], that products are almost always displayed in range of similar-look info pieces showing features and prices for customers to choose and so, the webpage DOM tree would have similar subtrees in the parts corresponding to the product show areas.**

We propose to use a special hash function, namely Simhash [18], for identifying the product regions. Our basic idea is that sub-trees (in the webpage DOM tree) with similar structures would have similar Simhash fingerprints (separated just by a few bits). To eliminate possible miscalls in the first phase using Simhash, we also combine with a decision tree approach which gives us more flexibility especially with product websites developed by Vietnamese companies which prefer certain display formats not very popular worldwide. Compared to PEWeb, our scheme can be more refined and flexible where we have more options to adjust the scheme. This improvement in preciseness is strongly supported by experimental results.

Keywords – Web data extraction, Product search, Simhash.

I. INTRODUCTION

Societies have been changing in many ways due to the emergence of the Internet, including our ways of shopping today which are heavily influenced by commercial websites and Internet-based services. However, due to the enormous number of commercial websites as well as web products and their vast information, customers have certain difficulties in searching for good products with affordable prices. Capturing this need, most popular Search Engines have their own searching services specializing in online products such as Google Product or Yahoo Shopping. These services aim at collecting, indexing and maintaining product information from the Internet, and hence, being able to respond users' queries more efficiently and effectively.

A major challenge that these product search services have to cope with is how to locate and extract correctly product description and related info from commercial websites. A commercial webpage normally comprises several display regions, such as menu, advertisement or product regions etc. A product extraction tool firstly must locate correctly the positions of product regions, before extracting from these areas specific info items about the products, such as title, price, image etc. This is not a trivial task since each website has its own layout, which may also be changed frequently, and moreover, the execution at each website must be fast enough because of a huge number of websites waiting to be processed. Currently, most available solutions are using by-hand or semi-automatic approaches which are costly and not highly durable.

There have been several approaches to the problem of product extraction, from using manual methods to semi or fully automatic ones. In manual methods, after observing a webpage and its source code, developers (programmers) consider a few samples and write a program to extract target data. Clearly, these methods are not practical enough for processing a large number of web pages. Wrapper induction is a popular semi-automatic method, proposed in the middle of 90's, where a set of rules are obtained from WebPages which are previously labeled by hand (STALKER [19], WIEN[14] and etc.). These rules are used to extract data on the similar web pages. Nevertheless, these tools are still time-consuming and highly costly

[i]A preliminary version of this paper appeared in a conference proceedings [38].

in building wrapper as well as labeling samples. Some tools including W4F [22], XWRAP[17] and RoadRunner [8] take advantages of inherent structural features of HTML documents, so can be more automated than wrapper induction tools. These, however, still requires the developer's work to create extraction rules and label samples. We leave a further review and analysis of the field in Section 2.

Most of the above extraction methods and tools are not fully automatic: they still have at least one phase which requires human intervention. This is definitely impractical on data sets with large size and frequent change such as commercial websites. Phan et al. [24], however, have proposed a simple yet efficient technique (PEWeb) for automatically locating product information from websites which possibly have different templates. They observed that the regions that describe products in a webpage usually have similar displays and hence, would result in similar subtrees in the DOM (HTML tag) tree of the webpage. Thus, they propose to use entropy estimation on the webpage DOM tree to identify these subtrees which have similar structures. They also use associate rules to filter out noise, i.e. regions that have a similar structure but are not product descriptions. PEWeb is fully automatic, however the quality of result is not impressive in experiments and entropy estimation is seemingly not a highly effective tool.

Products are always displayed in range of various features and prices for customers to choose. We also share the same main observation by PEWeb that web product regions are almost always displayed in arrays of adjacent pieces of areas of similar look and structure, and that the DOM HTML tag tree would have similar subtrees in the parts corresponding to the product show areas. However, we notice that using entropy can not always result in good results. High entropy scored by a tree node would indicate a high similarity between this node's subtrees. This applies to product regions but also applies to other array-based HTML objects which can be various in modern shopping websites such as menus, advertisements, category lists, etc. Thus an entropy-based filter can be in a dangerous situation of choosing a right threshold: it is narrow to go from accepting many noisy non-product objects to rejecting true product regions.

We propose to use a special hash function, namely Simhash [18], for picking out the product regions from the non-product. Simhash, as its name suggests, produces a similarity between hash values, i.e. fingerprints, of similar objects, while still runs very fast as other typical hash functions. Thus, we can assess a DOM tree (HTML tag) node as if presenting a product area by comparing fingerprints between any two of its subtrees. Compared to PEWeb, our scheme can be more refined and flexible where we have more options to adjust the scheme. In the second phase for filtering out the possible miscalls in the first phase, we use a decision tree approach instead of using association rules as in PEWeb. This also gives us more flexibility especially with product websites developed by Vietnamese companies which often use certain display format that can be more appearing to the Vietnamese customers.

The rest of the paper is structured as follows: after some quick review of related works in Section 2, we present the motivation and principles of our proposed techniques in Section 3, which also includes a detailed description and implementation of Simhash. Section 4 discusses a detailed implementation of the proposed techniques. Section 5 shows our results from experiments performed on real commercial websites (many are popular in Vietnam) which prove the effectiveness of our proposed techniques. Finally, we discuss concluding remarks in section 6.

II. RELATED WORKS

Web content extraction is an important phase of Web content mining that aims to extract useful and meaningful data out of webpages which are vast sources of noisy data such as ads and navigation links. The extracted content is crucial for many tasks such as information searching, natural language processing (NLP), information retrieval (IR), web documents classification, text translation, and etc. Because of its essential role, the problem of extracting meaningful data from the Web has been widely studied, resulting in many different useful approaches and algorithms. In this section, we present five different techniques for extracting Web content, and analyze their limits.

A. Wrappers for content extraction

Wrapper is a special program that extracts content from a particular information source and converts it into a structure. When generating a wrapper, two approaches are mainly used: wrapper induction [1] and automated data extraction. Wrapper induction applies supervised learning methods to learn the data extraction rules from manually labeled training data. Although many works [2],[3],[4] used this method to extract data from Web, wrappers have many shortcomings. First, it is not straightforward to construct wrappers as it requires expert users to write extraction rules, which is time-consuming and restricted. Moreover, when a page is changed, wrappers must be rebuilt, which makes the maintenance of wrappers very expensive. Two example tools of this type of extraction techniques are wrapper induction [1] and VEDD [4].

B. Template detection for extracting content

A template is a skeleton of HTML pages that is used as a basis to generate new webpages by plugging content into it. This results in collection of webpages having the same layout [5]. Many existing studies have introduced methods to detect the templates of webpages to extract useful content [5],[6],[9],[10],[11],[12]. The disadvantage of these results is that they heavily depend on the structures of webpages, which are not the same across different webpages. Worse still, the design of webpages is also frequently changed, which makes these methods not suitable when handling large number of webpages.

C. Content extraction using machine learning

Machine learning, an important branch of artificial intelligence, focuses on making predictions on unseen data, based on the properties learned from the trained data. Of many machine learning principles,clustering and classification are usually used in content extraction [13,15,16],[20,21,23]. For example, Debnath et al. [21] proposed a three-stage algorithm to find the primary informative content of webpages as follows: first, the algorithm classifies each of the document blocks extracted from a webpage as a redundant or identified block. Next, it extracts from the identified blocks some features like text, text-tag, list, etc. Finally, using a clustering algorithm, the blocks with desired features are considered the primary informative content. This method assumes that the primary content belongs to only one block, so it is not efficient for webpages with many main content areas. On the other hand, Pasternack and Roth[23] designed an extraction algorithm that is able to detect an article from webpages. In particular, after consecutive HTML segments that contain the webpage's article are identified, the algorithm removes unrelated segments and remaining segments form the article. As a supervised learning method, this approach requires to be retrained whenever a new feature of the web structure is added. Moreover, in some cases this method may falsely remove segments related to the article, making the output not acceptable.

D. Content extraction using visual cues and/or other assumptions

Severalexisting works for extracting content of webpages are based on visual cues and/or other certain assumptions [26],[27],[28],[29],[30]. In this approach, assumptions on the webpage's structure, content nodes, some kinds of tags, etc. are made to easy the extraction process. For example, Hong et al. [28] introduced Visual Wrapper for Extraction of Records (ViWER) algorithm to extract data using visual cues and DOM tree. ViWER first parses the Webpage into DOM tree and then extracts data based on visual cues and DOM tree properties. It uses four filter processes to detect and label the correct data regions. Meanwhile, Zhang and Wang [30] proposed a method for extracting meaningful information from Chinese webpages. The method consists of five steps, in most of which a lot of assumptions are made to help get meaningful content. These assumptions are made from some webpages so that they cannot be applied to others.

E. Content extraction based on HTML features and/or statistics

This is the mostly used and studied extraction technique as it gives better results than all above mentioned methods. Several methods using this extraction approach based on HTML features and/or statistics are proposed [31-37]. For example, Gottron[33] proposed a content extraction algorithm called Content Code Blurring (CCB). CCB represents the Webpage as a sequence (vector) of elements that are either code or content called content code vector. Then, regions which have the high ratio of content over code elements are useful, because they mainly contain content. However, this method does not differentiate between useful and useless content, and also is not applicable to multi-body webpages, *i.e*,

webpages containing multiple informative regions. Finally, Qureshi and Memon [36] designed a hybrid model to extract useful content by deriving from different methods. First, it converts HTML document to DOM tree, and then calculates link densities for each unique node in the tree. Consequently, nodes with the same styles and link densities are considered as useful contents. The shortcoming of this method is that it is too resource consuming and so, expensive. Furthermore, it may return some useless nodes if they have the same style as useful nodes, even though they have high link density.

III. PROPOSED TECHNIQUES FOR LOCATINGPRODUCT REGIONS

In this section, we describe our techniques for locating product regions from webpages. Firstly, we analyze a common layout of a commercial webpage. Then, we describeSimhash and its typical use, and present our Simhash-based techniqueto find regions in the webpages that contain potential product description. Finally, we will present ourdecision tree approach for the second, refining phase.

For simplicity, we only consider dealing with webpages which follow W3C HTML specification so that they can be parsed into DOM trees. Note that there exist non-standard webpages which do not conform to W3C specification, e.g. ones that have some structural errors due to bad coding (e.g. missing closing tags, overlapping paired tags, etc.). However in practice, a pre-processing phase can be used to remove all those structural errors and hence, DOM trees can still be built from these webpages.

A. Commercial webpages and their DOM trees

Figure 1 is a typical example of a commercial web-page. From the figure, it can be easily seen that description pieces for different products (laptops in this example) are displayed consecutively in an area (rectangle in dashed-line border). On the other hand, those product descriptions (in solid-lineborder) have the same layout: product images in the left, titles and further info in the middle, and prices in the right.

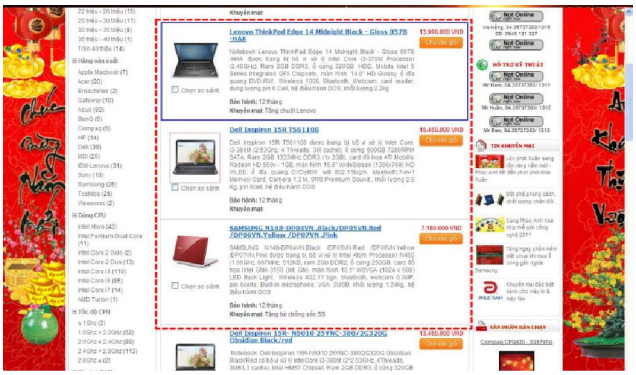

Figure 1. A web-page selling laptops

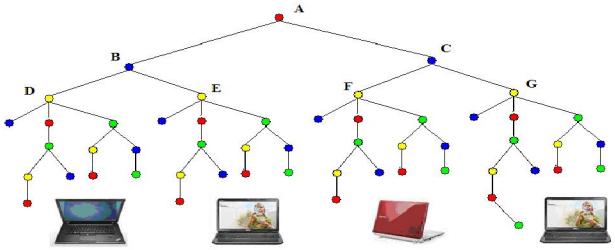

Figure 2. Product regions and their positions in DOM tree

If we convert HTML source code into a DOM (HTML tag) tree, the DOM tree has following properties: (i) Product regions correspond to the subtrees that are children of a node (because these regions lay consecutively in an area); (ii) subtrees of product regions have the same or similar structure. Figure 2 demonstrates these two properties. Four product regions correspond to four subtrees at the node D, E, F, and G. Nodes D and E have the same parent node B, F and G have the same parent node C. Subtrees at D, E, F have the same structure, while subtree G has a small difference from the others.

Thus, in order to identify product regions in webpages, we need to find subtrees in the DOM tree which satisfies two properties above. To do this, we must have a way to measure the similarity of subtrees. One of the existing solutions is using the tree edit-distance algorithm. Many versions of the tree edit-distance algorithm are proposed with improvements; however, their computational complexity is at least $O(|T_1||T_2|)$, where $|T_1|$ and $|T_2|$ are the sizes of two

trees. Therefore, the tree edit-distance cannot be a practical solution, observing that DOM trees usually have large sizes and also, the number of commercial websites is huge.

Below, we present Simhash as a simple and effective technique to measurethe structure similarity between DOM trees.

B. Simhash for detecting duplicate documents

Simhash was first introduced by Charikar [7]. It is a dimensionality reduction technique, which maps high-dimensional vectors to small-sized fingerprints. Then, Manku et al [18] has applied it to the detection of duplicate documents (webpages) crawled by Search Engines. This detection mechanism is very important for Search Engines, because its success will enhance the search results' quality.

In duplicate document detection, Simhash is used to generate a fingerprint for a web-page as follows. Firstly, we convert a web-page into a set of features, each of which has its weight. An f-dimensional vector V is then initialized where each dimension is set to zero. A feature is then hashed into an$f$-bit fingerprint. These f bits have their crucial influence on vector V as follows: if the i-th bit of the fingerprint is 1, the i-th component of V is increased by the weight of that feature; otherwise(bit 0), the i-th component of V is decreased also by thatamount. Finally, having all features processed and contributed to V,the f-bit fingerprint of the web document is constructed as follows: the i-th bit's value (1 or 0) in this fingerprint is determined by the sign(plus or minus) of the corresponding i-th componentin V.

Clearly, this Simhashfunction has a special property, never seen in previous hash functions: similar documents do have similar fingerprint, i.e. their fingerprints differ in few positions. Therefore, to measure the similarity between two documents, we just need to identify the number of bit positions which does not have the same values in their fingerprints.

C. Simhash for similar subtree search on a DOM tree

The idea of using Simhashto identify similarity between documents can be applied onto subtreesof a webpage DOM tree. Each subtree in the DOM tree will be *Simhashed*to a fingerprint. The number of different bit positions between two fingerprints measures the difference or the similarity between 2 subtrees. An important feature of DOM trees is that they are labeled tree where nodes have labels, i.e. HTML tag names. Therefore, two similar DOM trees would have similar structure and also their nodes in the same positions would have the same labels.
Having obtained fingerprints of all children nodes of a tree T, we compute the tree T's fingerprint in steps as follows. We also maintain an f-bit dimensional vector V with each component set to zero. Each child node's fingerprint would increments or decrements the V's components as follows: if the i-th bit of the fingerprint is *1*, then the i-th component of V is added by *the size of the subtree* at that child node (or the number of nodes in the subtree), otherwise this addition is replaced by subtraction. As in Section III-B, we compute an f-bit value based on the signs of V's components: 1 for plush, 0 for minus. Finally, this f-bit number is XORedwith the f-bit hash value of T's label to form T's fingerprint. An example of a Simhash calculation process for a DOM tree is described in Fig. 3.

There are some notes from the above Simhash function: (1) A node's fingerprint is computed from its child nodes' fingerprints; therefore, the tree Simhash computing process is bottom-up, (2) The effect of each node child on vector V depends on its subtree's size, because we assume that the larger the subtree is, the more important it is and the more "contribution" it makes to the final value of V, (and the fingerprint later), (3) In the final step, the tree's fingerprint is constructed from f-bit number made from vector V and the hash value of tree's label, as we consider that two similar DOM trees must first have the same root's label. This is true in the case of trees representing product regions when we observed online commercial websites.

D. A decision tree for filtering process

With the Simhash function for trees, we can find all the subtrees in the DOM tree each of which has a structure similar to at least another subtree. However, not all the resulted subtrees actually correspond to product regions. Some of them correspond to advertisements, menu, navigation links, etc. They are considered as noise, and we have to remove them.

PEWeb gives a score (product possibility) to each tree nodeby using a set of association rules, which is achieved from a training process with a set of tree nodes in different commercial websites. A score

threshold is used to decide whether a tree node is outputted as corresponding to a product description.

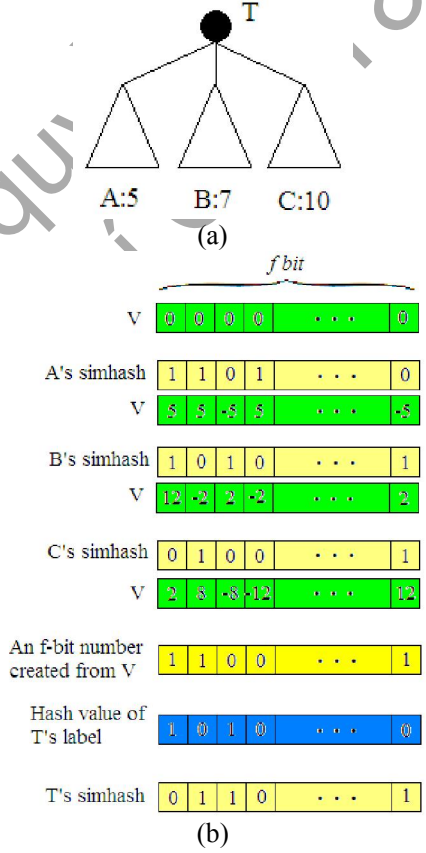

Figure 3. An example of Simhash calculation in tree at node T. (a) Tree at node T has 3 subtrees: A, B and C, which have sizes of 5, 7 and 10, respectively. (b) Simhash calculation process for T-rooted tree.

A disadvantage of PEWeb's filtering technique is that its accuracy depends on the score threshold. This threshold varies in different websites and hence, it is practically hardto pick a common threshold for all websites. In our system, we use an alternative technique to filter out noisy results. We use a decision tree for this filtering process. The decision tree is built from a training set of *1000* records with *11* attributes. These records correspond to subtrees collected from many different commercial websites. The training set includes both positive (product regions) and negative (noisy regions) samples.

The *11* attributes that are chosen based on some classification heuristics, are listed in the Table *1*.

To build the decision tree from the training set above, we use WEKA [25], a popular suite of open source machine learning software, and the result we obtain is shown in Figure 4.

Table 1. Attributes of each records for the filtering process

Attributes	Description
nLinks	Number of <a> tag
nImages	Number of <img> tag
nFonts	Number of <font> tag
nBolds	Number of <b> tag
nItalics	Number of <i> tag
nBrs	Number of tag
nLists	Number of tag
nUnits	Number of currency signs (such as VNĐ, USD, $, …)
nPrices	Number of words "Giá", "giá", "Price", …
nPromos	Number of words "khuyến mại", "khuyến mãi"…
nDigits	Number of digital characters

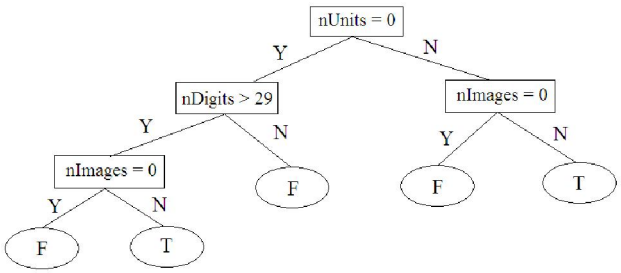

Figure 4. The decision tree determining whether a region is a product region or not

As can be easily seen in Fig. 4, our decision tree does not use all the attributes we have chosen. Actually, it takes just 3 attributes: nUnits, nImages and nDigits to decide if a region is product region. It can also be concluded from the decision tree that, among these 3 attributes, nUnits is easily the most important attribute. Indeed, in our training set, most of the product regions have the nUnits attribute greater than 0, which makes it substantial information for classification. Clearly, in the real commercial websites product descriptions usually contain prices (with currency signs).

IV. IMPLEMENTATION

This section describes how to apply our proposed techniques above to build our product information extraction system. The extracting process for a webpage includes two phases: finding potential regions in the webpage and filtering noise (wrong picks from the first phase).

A. Finding potential product regions

As we said earlier, finding regions which may be product regions is actually finding all the similar subtrees in DOM tree using Simhash. The pseudo-code of Simhash algorithm is described in Algorithm 1. The algorithm receives as input a T-rooted tree and will calculate the fingerprint of node T.

After the first traversal over the webpage DOM tree to calculate the fingerprints of all the nodes, another traversal is required to find similar subtrees. This function findProductRegions is shown in Algorithm 2 below. Two subtrees will be in the result set if they satisfy the following conditions:

- They all belong to one parent node.

- Their depth is greater than h, a system parameter.

- Their fingerprints differ in at most n bit-position (n is called the difference threshold, another system parameter).

The essence of the first condition was explained in the previous section: product regions are usually displayed consecutively in an area, which corresponds to subtrees having the same parent node. The second condition is suggested from our observations: product regions usually reside in subtrees with depth greater than a given depth threshold, i.e. parameter h. Thus, we use this threshold to remove "noisy" subtrees which do not actually correspond to product regions.

Algorithm 1 - Simhash : Simhash Fingerprint Calculation

Input: node T in DOM HTML tree

Output: T's Simhash fingerprint

Initialize 32-dimensional vector V, with all components set to 0

for each child node t of T **do**
  for i from 0 to 31 **do**
    if t's i-th bit is 1 **then** V[i] = V[i] + t.num_node
    else V[i] = V[i] - t.num_node

Create a number H from V as follows:

for i from 0 to 31 **do**
  if V[i] >= 0 **then** H's i-th bit is set to 1
  else H's i-th bit is set to 0.

T.fingerprint = H XOR hash(T.tag_name)

Algorithm 2 – findProductRegions : Similar subtrees (or potential product regions) search.

Input: node T in the DOM HTML tree

Output: all similar-structured subtrees of T.

check = False
for each child node t_1 of T **do**
  if t_1.depth > h **then**
    for each child node t_2 of T **do**
      if $t_1 \neq t_2$ and t_2.depth > h and diff(t_1.Simhash, t_2.Simhash) < n **then**
        productRegionSet.add(t_1)
        productRegionSet.add(t_2)
        check = True
if check = False **then**
  for each child node t_1 of T **do**
    if t_1.depth > h **then**
      findProductRegions(t_1, productRegionSet)

The input of function findProductRegions is a node T' and the output is a set of subtrees, productRegionSet, of a tree rooted at T satisfying the above conditions. This is a recursive function, whose base case is when the T's size is too small (its depth is smaller than h) or T's child nodes are product regions.

B. Filtering noise

From the decision tree in Section III-D, we have the following checking function. The function receives a region (a piece of HTML source code) as an argument, and returns *True* if it is a product region, *False* if not. Our filtering function is simple, compared to that of PEWeb system. Our function needs just at most *3 comparisons* to identify the product region. Meanwhile, in PEWeb system, a region has to be checked through *30 condition* expressions, with at least *3 comparisons* in each expression. This has a major impact to PEWeb's performance when applying to large data sets.

Algorithm 3 - isProductRegion(R): Product region checking.

Input: a region in the webpage.

Output: *True* if it is a product region, *False* otherwise.

```
if R.nUnits = 0 then
    if R.nDigits > 29 then
        if R.nImages = 0 then return False
        else return True
    else return False
else
    if R.nImages = 0 then return False
    else return True
```

V. EXPERIMENTAL RESULTS

In this section, we show experimental results produced with our system which implementsour proposed techniques. We also compare our system to PEWeb system which solves the same problem, based on using entropy estimation.

Data: Our testing data isa selected collection of webpages from popular Vietnamese commercial websites. These are about various kinds of products,such as computers, mobile phones or other appliances. Each website often has few different templates and so we just pick one or two webpages pereach template, i.e. because working with webpages using the same template will just give the same performance.

Parameters: In our system, there are 2 important parameters: difference threshold n and the minimum depth h. The setting of n plays a crucial role to the Simhash's performance, because if n is too large, there may be many noise results from the searching process, if n is too small, we may miss correct results. After testing various values of n in webpages, we set n equal to 4 as a reasonable value. Parameter h is set to 3. In PEWeb system, there are 3 options: *minimum depth threshold (DTh), minimum entropy ration threshold (ERTh),* and *minimum score threshold (STh)*. The defaults of these parameters are 3, 0.90, and 15 respectively. In our experiments, we keep this default value for these options.

Experimental results. The experimental results are shown in Table 2. The first column is the table index while the second column is the URL of a website. The third column shows the total quantity of products in the pages sampled from the corresponding website. The next two columns are on the outputs generated by our Simhash-based system for these particular pages: the first of the two is the number of product items found by the system and the second is the number of correct picks. The last two columns show the same two output quantities for PEWeb system. In the bottom of table, we give the corresponding sums of all the statistics above. Then, the two measures *recall* and *precision* of two systems are calculated for all the websites.

Discussion. It is easy to seefrom Table 2 that our system outperforms PEWeb in terms of precision and completeness. The recall of our system, 98.2%, is higher than that of PEWeb system, 88.5%, which demonstrates that our Simhash-based technique is more effective than PEWeb's entropy-based one for finding similar subtrees. It seems that the PEWeb's performance is much influenced by the score threshold, the parameterto be set the same for all websites, and that is a hard task of picking the right value. Our precision score (99.8%) is also higher than that of PEWeb (93.1%) because our second-phase filtering process works better than the one in PEWeb. It is also understandable, because our filtering method (using a decision tree) just focuses on the most typical features of product regions (number of current signs, images, etc.), while the heuristic features used by PEWeb maybe partially redundant.

However, note that in some websites, the results of both systems are poor, because these websites use special scripts to construct the ultimate HTML code, such as http://vatgia.com/, http://laptopcaocap.com/, http://www.dienmaycholon.vn/, etc. To get the whole ultimate HTML code, we need a proper script-engine to run this script (and analyze the obtained code). This extra step is beyond the scope of ours aim at the present, but would be considered and achieved in a future extension of our system.

Although our system has just focused on Vietnamese commercial websites, we can easily expand it to any other website worldwide, simply by extending the training set in the filtering process to cover those.

VI. CONCLUSION AND FUTURE WORKS

This paper proposes an effective yet simple technique for automatically locating regions of product information from a commercial webpage. Our technique uses a special hash function, namely Simhash, for identifying the potential product regions and a decision tree-based filter for eliminating wrong choices made in the first phase. Using Simhash is particularly suitable and efficient for finding subtrees

that have similar structures in a webpage DOM tree: Simhash fingerprints are so fast to generate and compare. The use of a decision tree for removing incorrect outputs in the filtering second phase will enhance the quality of the whole process. Our experiments results prove a strong improvement over PEWeb, the previous system which tackles the same problem of locating production regions.

In the future, we will further study the problem of extracting product information from webpages. The next step after the task of automatically locating the product regions will be locating and extracting values of the required fields from the obtained regions, such as product title, images, prices, and other relevant information fields. We suggest to do that by using some specific features of commercial websites related to the semantics, vocabulary, and HTML tags etc., which would be typical there. Moreover, we also intend to extend our overall system (in extracting product info from the Web) so that it can automatically identify commercial websites from others. This will improve the automation level of our system which is very important when dealing with such a huge data source, the Internet.

REFERENCES

[1] Kushmerick, N. 1997. Wrapper induction for information extraction. Doctoral dissertation, University of Washington.

[2] Bharanipriya, V., & Prasad, V. K. 2011. Web content mining tools: a comparative study. International Journal of Information Technology and Knowledge Management, 4(1), 211-215.

[3] Ling, L. I. U., PU, C., & Wei, H. A. N. 2000. XWRAP: an XML—enable wrapper con—struction system for the Web information source. In Proc of the 16th IEEE International Conference on Data Engineering . Vol. 611, p. 620.

[4] Tripathy, A. K., Joshi, N., Thomas, S., Shetty, S., & Thomas, N. 2012. Vedd-a visual wrapper for extraction of data using dom tree. In Communication, Information & Computing Technology (ICCICT), 2012 International Conference on (pp. 1-6). IEEE.

[5] Bar-Yossef, Z., & Rajagopalan, S. 2002. Template detection via data mining and its applications. In Proceedings of the 11th international conference on World Wide Web (pp. 580-591). ACM.

[6] Chakrabarti, D., Kumar, R., & Punera, K. 2007. Page-level template detection via isotonic smoothing. In Proceedings of the 16th international conference on World Wide Web (pp. 61-70). ACM.

[7] Charikar, M. 2002. Similarity estimation techniques from rounding algorithms. In Proc. 34th Annual Symposium on Theory of Computing (STOC 2002), pages 380-388.

[8] Crescenzi, V., Mecca, G. and Merialdo, P. 2001. Roadrunner: Towards automatic data extraction from large web sites. Proceedings of the 26th VLDB, pages 109-118.

[9] Chen, L., Ye, S., & Li, X. (2006, April). Template detection for large scale search engines. In Proceedings of the 2006 ACM symposium on Applied computing (pp. 1094-1098). ACM.

[10] Hong, J. L., & Fauzi, F. 2010. TreeWrap-Data Extraction Using Tree Matching Algorithm. Majlesi Journal of Electrical Engineering, 4(2), 43-55.

[11] Lei Fu, Yao Meng, YingJu Xia, Hao Yu. 2010. Web Content Extraction based on Webpage Layout Analysis. Information Technology and Computer Science (ITCS),pp.40-43.

[12] Wang, Y., Fang, B., Cheng, X., Guo, L., & Xu, H. 2008. Incremental web page template detection. In Proceedings of the 17th international conference on World Wide Web (pp. 1247-1248). ACM.

[13] Kao, H. Y., Lin, S. H., Ho, J. M., & Chen, M. S. 2004. Mining web informative structures and contents based on entropy analysis. Knowledge and Data Engineering, IEEE Transactions on, 16(1), 41-55.

[14] Kushmerick, N. 2000. Wrapper induction: Efficiency and expressive. Artificial Intelligence, 118(1-2):15-68.

[15] Li, Z., Ng, W. K., & Sun, A. 2005. Web data extraction based on structural similarity. Knowledge and information systems, 8(4), 438-461.

[16] Weninger, T., Hsu, W. H., & Han, J. 2010. CETR: content extraction via tag ratios. In Proceedings of the 19th international conference on World wide web (pp. 971-980). ACM.

[17] Liu, L., Pu, C. and Han, W. 2000. Xwrap: An xml-enable wrapper construction system for web information sources. Proceedings of the 16th IEEE International Conference on Data Engineering, pages 611-612.

[18] Manku, G. S., Jain, A. and Sarma, A. D. 2007. Detecting near-duplicates for web crawling. In Proceedings of the 16th International Conference on World Wide Web, pp. 141-150 Banff, Alberta, Canada.

[19] Muslea, I., Minton, S. and Knoblock, C. A. Hierarchical wrapper induction for semistructured information sources. Autonomous Agents and Multi-agent. 4(1-2):93-114, 2001.

[20] Zachariasova, M.; Hudec, R.; Benco, M.; Kamencay, P.2012.Automatic extraction of non-textual information in web document and their classification.

Telecommunications and Signal Processing (TSP), pp.753-757.

[21] Debnath, S., Mitra, P., & Giles, C. L. 2005. Identifying content blocks from web documents. In Foundations of Intelligent Systems (pp. 285-293). Springer Berlin Heidelberg.

[22] Sahuguet, A., and Azavant, F. Building intelligent web applications using lightweight wrappers. Data and Knowledge Engineering 36, 3, 283-316.

[23] Pasternack, J., & Roth, D. 2009. Extracting article text from the web with maximum subsequence segmentation. In Proceedings of the 18th international conference on World wide web (pp. 971-980). ACM.

[24] Phan, X.H., Horiguchi, S. and Ho, T.B. 2004. PEWeb: product extraction from the web based on entropy estimation. The IEEE/WIC/ACM Conf. on Web Intelligence, IEEE Computer Society 20-24, Beijing, China, pp.590-593. DOI= http://dx.doi.org/10.1109/WI.2004.114.

[25] WEKA. Data Mining with Open Source Machine Learning Software in Java. http://www.cs.waikato.ac.nz/ml/weka/.

[26] Lin, S. H., & Ho, J. M. 2002. Discovering informative content blocks from Web documents. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 588-593). ACM.

[27] Cai, D., Yu, S., Wen, J. R., & Ma, W. Y. 2003. Extracting content structure for web pages based on visual representation. In Web Technologies and Applications (pp. 406-417). Springer Berlin Heidelberg.

[28] Hong, J. L., Siew, E. G., & Egerton, S. 2010. ViWER-data extraction for search engine results pages using visual cue and DOM Tree. In Information Retrieval & Knowledge Management,(CAMP), 2010 International Conference on (pp. 167-172). IEEE.

[29] Yang, D., & Song, J. 2010. Web content information extraction approach based on removing noise and content-features. In Web Information Systems and Mining (WISM), 2010 International Conference on (Vol. 1, pp. 246-249). IEEE.

[30] Bin ZHANG, Xiao-fei WANG. 2012. Content extraction from Chinese web page based on title and content dependency tree. The Journal of China Universities of Posts and Telecommunications, Volume 19, Supplement 2, Pages 147-151.

[31] Gupta, S., Kaiser, G., Neistadt, D., & Grimm, P. 2003. DOM-based content extraction of HTML documents. In Proceedings of the 12th international conference on World Wide Web (pp. 207-214). ACM.

[32] Pinto, D., Branstein, M., Coleman, R., Croft, W. B., King, M., Li, W., & Wei, X. 2002. QuASM: a system for question answering using semi-structured data. In Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries (pp. 46-55). ACM.

[33] Gottron, T. 2008. Content code blurring: A new approach to content extraction. In Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on (pp. 29-33). IEEE.

[34] Adam, G., Bouras, C., & Poulopoulos, V. 2009. CUTER: An efficient useful text extraction mechanism. In Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on (pp. 703-708). IEEE.

[35] Mingsheng, H., Zhijuan, J., & Xiangyu, Z. 2012. An approach for text extraction from web news page. In Robotics and Applications (ISRA), 2012 IEEE Symposium on (pp. 562-565). IEEE.

[36] Qureshi, P. A. R., & Memon, N. 2012. Hybrid model of content extraction. Journal of Computer and System Sciences, 78(4), 1248-1257.

[37] Insa, D., Silva, J., & Tamarit, S. 2013. Using the words/leafs ratio in the DOM tree for content extraction. The Journal of Logic and Algebraic Programming, 82(8), 311-325.

[38] Pham, T. A. N., & Nguyen, V. K. 2011. A simhash-based scheme for locating product information from the web. In Proceedings of the Second Symposium on Information and Communication Technology (pp. 199-206). ACM.

AUTHORS' BIOGRAPHIES

Tuan-Anh N. Pham is a Ph.D. student from School of Computer Engineering (SCE), Nanyang Technological University (NTU), Singapore. He received his Bachelor (2009) and Master (2011) degrees in School of Information and Communication Technologies (SoICT) of Hanoi University of Science and Technology (HUST). His research interests include data mining and information retrieval.

Khanh-Van Nguyen is a Senior Lecturer at the School of Information and Communication Technologies (SoICT) of Hanoi University of Science and Technology (HUST), where he leads the Software Engineering and Distributed Computing Lab (SEDIC Lab). He received his Ph.D. degree in Computer Science from the University of California, Davis (2006) and also spent one post-doctoral year (2007-08) as a CNRS researcher at University Paris 7 & 11. His main research domain is algorithms and theoretical models.

	Websites	Number of products	Simhash		PEWeb	
			Number of outputs	Number of correct outputs	Number of outputs	Number of correct outputs
1	http://phucanh.vn/	35	26	25	10	4
2	http://trananh.vn/	115	115	115	96	96
3	http://www.vatgia.com/	47	47	47	47	47
4	http://nhatcuong.com/	20	18	18	13	13
5	http://www.thegioididong.com/	20	7	7	20	20
6	http://clickbuy.com.vn/	91	91	91	100	89
7	http://viettelstore.vn/	55	53	53	18	18
8	http://www.mainguyen.vn/	53	40	40	48	48
9	http://cellphones.com.vn/	45	37	37	16	16
10	http://didongviet.vn/	27	21	21	25	24
11	http://www.sieuthicomputer.com.vn/	137	136	136	141	137
12	http://www.hanoicomputer.vn/	11	51	49	72	39
13	http://www.nguyenkim.com/	33	32	32	23	23
14	http://www.tabletworld.vn/	24	20	20	10	9
15	http://cellphones.com.vn/	45	37	37	16	16
16	http://didongviet.vn/	21	21	21	21	21
17	http://dienmaycholon.vn/	40	39	39	32	32
18	http://www.nguyenkim.com/	30	30	30	30	30
19	http://pico.vn/	91	91	91	90	90
20	http://mediamart.vn/	86	85	85	85	85
21	http://www.topcare.vn/	32	30	25	32	32
22	http://www.dienmay.com/	18	18	18	18	18
23	http://www.123mua.vn/	51	50	50	43	43
Total		1127	1095	1087	1006	950
Precision			**0.992694064**		**0.944333996**	
Recall			**0.964507542**		**0.842945874**	

Table 2. The experimental results of both systems