Data Extraction for Search Engine Using Safe Matching

Jer Lang Hong[1], Ee Xion Tan[2], and Fariza Fauzi[2]

[1] School of Computing and IT, Taylor's University
jerlang.hong@taylors.edu.my
[2] School of IT, Monash University
{tan.ee.xion,wan.fariza}@monash.edu

Abstract. Our study shows that algorithms used to check the similarity of data records affect the efficiency of a wrapper. A closer examination indicates that the accuracy of a wrapper can be improved if the DOM Tree and visual properties of data records can be fully utilized. In this paper, we develop algorithms to check the similarity of data records based on the distinct tags and visual cue of the tree structure of data records and the voting algorithm which can detect the similarity of data records of a relevant data region which may contain irrelevant information such as search identifiers to distinguish the potential data regions more correctly and eliminate data region only when necessary. Experimental results show that our wrapper performs better than state of the art wrapper WISH and it is highly effective in data extraction. This wrapper will be useful for meta search engine application, which needs an accurate tool to locate its source of information.

Keywords: Information Extraction, Automatic Wrapper, Search Engines.

1 Introduction

A computer user is able to obtain relevant information from the World Wide Web simply and quickly due to the advent of Information Technology. As the World Wide Web contains a huge amount of data, the extraction of required information has been significantly simplified as the user needs to enter only search queries for the database servers to generate the information needed and deliver directly to the user. The generated information is usually enwrapped in HTML (HyperText Markup Language) pages as data records and it forms the hidden web (or deep web or invisible web). As the generated data records from the deep web is highly dynamic, it is difficult for the current search engines to index these HTML pages. Thus, these web pages are called deep web pages. Data records presented in a web page are usually presented using a predefined template, and these data records normally possess similar structures and patterns to form groups of data called data region. Specific information such as relevant data records from the deep web pages are the main source of information for a meta search engine. However, before data records can be used in a meta search engine, they need to be extracted from the search engine results page and converted to a machine readable form. Automatic wrapper is the tool developed for this purpose and it is used to automate meta search engine to increase the speed and efficiency of these search engines [1], [2].

D. Wang and M. Reynolds (Eds.): AI 2011, LNAI 7106, pp. 759–768, 2011.

A meta search engine normally receives a user's query and disambiguate the query for further processing. The query will then be passed to other search engines after they are classified based on the search query. The search engines will generate search results based on the query and this needs a tool (wrapper) to extract the information before it is sent back to the original server. Meta search engine will then compile the returned information, filter out the irrelevant search results, rank them and display the final results to the user. An accurate and fast tool is required to extract the relevant information from search engine results pages so that the meta search engines can be more efficient in ranking and filtering the search results in a timely manner. Thus, the robustness and accuracy of a wrapper will greatly affect the performance of a meta search engine.

Current wrappers use DOM Tree and Visual Cue to extract relevant data region from search engine results pages. These wrappers use the regularity of the structure and layout of data records for data extraction. Our observations show that 30% of the relevant data regions contain irrelevant information such as search result identifier (*e.g. Search returns 10 records*). Current wrappers (MDR[1], DEPTA[8], WISH[4]) are used to extract data records in a sequential order (first to last data records) and they are not designed to extract the said data regions as these data regions contain a mixture of similar and dissimilar data records. In this paper, we propose a novel and robust wrapper to extract relevant data region from search engine results pages. We use algorithms which are able to check the similarity of data records more accurately. Our approach is to use algorithms to check the similarity of data records using the DOM Tree and visual cue properties of these records which are unique and can be recognized easily. In order not to exclude data regions unnecessarily, we also develop a voting algorithm to distinguish the relevant data region, that is, if 85% of its data records are similar, the data region will be treated as potential relevant data region and retained for further processing. Our wrapper then extracts the relevant data region from the list of available data regions. Finally, we use a filtering technique to remove irrelevant data (search identifiers) from the relevant data region. Our wrapper is called **SafeMatch**ing Wrapper (**SafeMatch**).

Our wrapper is divided into 2 components. We first discuss the data extraction module of our wrapper. Given a web page, our approach is to parse the page and arrange it into a DOM Tree. We then use an Adaptive Search Algorithm to label and detect the correct data region. Our algorithm uses a few filtering stages which are able to group the list of data records available in a web page and filter out irrelevant information such as menu bars. Potential data records are then passed through the similarity check and data record detection filters to further exclude irrelevant data to obtain the correct data region. Tests on datasets obtained from various sources and comparison with other existing wrappers show that our wrapper is highly effective in data extraction.

This paper contains several sections. Section 2 describes the current work that is related to ours. Section 3 gives the implementation details of our wrapper. Section 4 provides the experimental tests conducted on our wrapper and finally Section 5 summarizes our work.

2 Related Work

DEPTA [8] uses a bottom up tree matching algorithm to match tree structures of data records. A tree matching algorithm matches two tree structures and determines how

the first tree can be transformed into the second tree. DEPTA's tree matching algorithm determines the maximum matches between two trees by comparing the location and identity of the nodes in the tree structures. DEPTA checks the similarity of two trees using the percentage similarity of the trees.

ViNT [3] extracts content line features from the HTML page, where a content line is a type of text which can be visually bounded by a rectangular box. Content lines are categorized into 8 types, each with their own distinguishing characteristics and features, which are grouped to form content blocks. ViNT parses these content blocks to identify the data records. Essentially, ViNT defines a data record as a content block containing a specific ordering of content lines.

ViPER [5] takes a more "natural" approach by projecting the contents of the HTML page onto a 2-dimensional X/Y co-ordinate plane, effectively simulating how the HTML page may be rendered on a printed hard-copy. This enables ViPER to compute two content graph profiles, one for each X and Y planes, which it uses to detect data regions by locating valleys between the peaks as the separation point between two data records (valleys are usually the space within two data records, separating them apart).

WISH [4] uses frequency measures to match the tree structures of data records. WISH works in a time complexity of O(n) and is able to match tree structures of data records containing iterative and disjunctive data. However, tree matching algorithm of WISH is not able to match data records with dissimilar tree structures.

Recently, ODE wrapper [7] uses ontology technique to extract, align and annotate data from search engine results pages. However, ODE requires training data to generate the domain ontology. ODE is also only able to extract a specific type of data records (single section data records), thus it is not able to extract irregular data records such as multiple sections data records and loosely structured data records.

3 SafeMatch Wrapper

3.1 Overview of SafeMatch

SafeMatch requires that the HTML web page of the search engine result pages are parsed and stored in a DOM Tree. In order to simplify our wrapper, we assume that the page under extraction must contain at least 3 repetitive patterns. This assumption is based on our observations that the majority of search engine result pages contain more than 3 repetitive patterns. These useful repetitive patterns are potential data records. SafeMatch consists of two main components. The first component involves parsing the HTML page and organizing it into Document Object Model (DOM) tree representation. In the second component, SafeMatch extracts data records using visual cue and DOM properties of the data records. In Component 2, SafeMatch goes through four stages for the extraction of data records. The initial stage is to come out with a list of data records. The initial list before the filtering processes is usually large. At each step from Stages 1 to 4 of the filtering processes, SafeMatch reduces the list by removing irrelevant data records in each data region. The underlying implementation varies in each of these stages and will be explained in detail in Section 3.2. At the end of the filtering processes, if successful there will only be one data region with only the correct data records left.

3.2 SafeMatch Extraction Module

Overview

Once the DOM tree is constructed, it is passed through a three filtering stages to filter out irrelevant information and identify the correct data region which also contains the data records. Before the filtering processes can be carried out, we use the Adaptive Search Algorithm to detect and label potential data records in a DOM Tree. Details of these works are presented in the following sections.

Assumptions

We have made several important observations on several unique features inherent to a data record. Based on these observations, we come out with a way to correctly extract data records. The following are the observations made by us:

Observation 1
The size of the data records in a search engine results page is usually large in relation to the size of the whole page.

Observation 2
Data records usually occur three or more times in a given search engine results page.

Observation 3
Data records usually conform to a specific regular expression rule to represent their individual data, hence they have nearly similar tree structure.

We examine carefully these three observations and found that these criteria could be formulated using visual cue and DOM tree structure of data records. Three steps of filtering rules are proposed, each of them considering the above observations.

Adaptive Search

The inclusion of this stage is to detect and label the different groups of potential data records. Groups of data records can be defined as a set of data records having similar parent HTML tag, containing repetitive sequence of HTML tags and are located in the same level of the DOM tree. SafeMatch uses the Adaptive Search extraction technique to determine and label potential tree nodes that represent data records. Subtrees which store data records may be contained in potential tree nodes. The nodes in the same level of a tree are checked to determine their similarity (whether they have the same contents). If none of the nodes can satisfy this criterion, the search will go one level lower and perform the search again on all the lower level nodes. Our method involves the detection of repetitive nodes which may contain data records and the rearrangement of these nodes to form groups of potential records in a list in 2 steps:

1. In a particular tree level, if there are more than 2 nodes and a particular node occurs more than 2 times in this level, SafeMatch will treat it as a potential data record irrespective of the distance between the nodes.

2. These potential data records identified in this tree level are then grouped and stored in a list. The potential data records in this list are identified by the notation $[_{A1}, _{A2}, \ldots _{An}]$ where $_{A1}$ denotes the position of a node in the potential data records where it first appears, $_{A2}$ is the position where the same node appears the second time and so on. Fig. **1** shows an example where nodes A, B, C are grouped and stored in list 1.

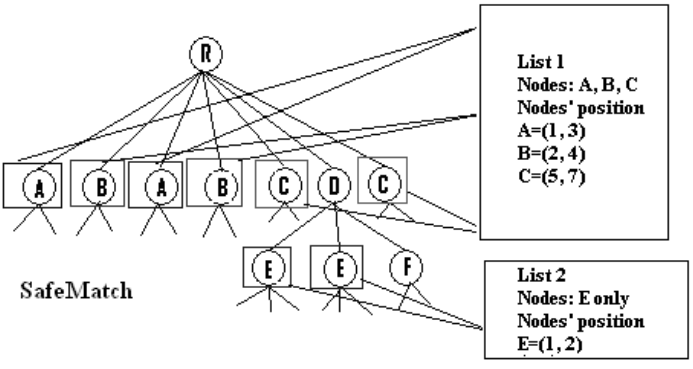

Fig. 1. Potential data records in SafeMatch

Overview of SafeMatch Extraction Rules

After going through the Adaptive Search stage, SafeMatch will have a list of data regions. Our examination shows that data regions fall into one of several groups. We group the first set of potential data regions as menus, this group of data regions determines the layout of HTML pages and is usually large in size and highly dissimilar. The second group is advertisements, regions of this group are highly similar but with simple structures. The third group consists of menu bars, these regions are simple but nearly similar in structure. It is the last group of data records that are relevant to our work, the search engine results output, these regions are highly similar in structure and large in size. We aim to design our wrapper so that it can extract this last group of data regions, while removing the other irrelevant ones. We used filtering stage 1 to remove menus which determine the layout of the HTML page, and filtering stage 3 to remove the remaining irrelevant data (e.g. advertisements). Filtering stage 2 is designed to remove data records which occur less frequently, as observed by author of [8].

Stage 1: Similarity Filter

Proposed Algorithms

In this section, we introduce our algorithms which are able to check the similarity of data records more accurately. Our algorithms include the DOM tree based and visual cue similarity check and a voting algorithm. We derived the DOM tree based matching algorithm based on Observation 3 and our finding that data records share an

important characteristic, i.e. the distinct tags of a tree and the total number of distinct tags in each level of the tree are nearly similar to those of the other trees of the group. Thus we are able to formulate a similarity check algorithm which can mimic the behavior of a full tree matching algorithm. Our approach is to carry out the similarity check of two trees by examining the distinct tags and comparing the total number of distinct tags in all levels of the trees. Our algorithm is simple but efficient and it can obtain similar results as those of a tree matching algorithm but it has a reduced time complexity. Our further investigation shows that visual information of data records is also useful in checking the similarity of these data records. This can be achieved by comparing the sizes of the bounding boxes which contain the data records. As noted, not all the data records in the relevant data region are similar, for example, some pages contain relevant data region with search identifiers. It is considered appropriate that if 85% of the data records in a data region are similar, this data region can be treated as relevant and retained for further processing. Otherwise, the data region will be discarded. Our voting algorithm is designed to check whether more than 85% of data records in a data region are similar. It is used together with the DOM Tree and visual cue algorithms to check the similarity of data records. Our similarity check works as follows:

1. The tree structure of data records in a data region are first checked for similarity using the DOM Tree based algorithm. If they pass the test, the voting algorithm will be used to further check the similarity of the data records (i.e. whether more than 85% of the data records are similar)
2. If the trees in the data records are similar, then the trees of Item 1 will be used for visual similarity check and if they pass the test, the voting algorithm will be applied for further test and if they pass the test again, they are considered similar.

The details of our algorithm and its use in detecting similarity of data records and filtering dissimilar data regions are presented in the following subsections.

DOM Tree Based Similarity Check

Our Tree Matching algorithm consists of a two stage screening procedure to check the similarity of a group of trees. Given a number of trees, our algorithm first examines the distinct tags of the first tree and those of the second tree. If almost all the distinct tags occur concurrently in the two trees (overall with say only one element different), then the trees pass the similarity test of the first stage and they are used for the second stage similarity test. In the second stage, we calculate the total number of distinct tags in all the levels of the first tree and that of the second tree. If the first two trees have almost equal number of distinct tags in all levels of the trees (overall with a difference of only one tag), then the two trees are considered similar according to the stage two criterion. The first two trees are similar only if they pass the screening procedures of both stages. If the first two trees are similar, the first tree is retained for further processing and the second tree is then compared with the third tree of the group to check their similarity using Stages 1 and 2 of our screening algorithm. On the other hand, if the first two trees are not similar, our voting algorithm will mark this data region as having one dissimilar data record and the second tree will be compared with the third tree to check their similarity. The screening procedures for both the above cases are repeated until the last tree is used for comparison.

Fig. **2** shows data records presented in a tree form obtained from the DOM Tree of HTML pages. For simplicity, we show only two trees in each figure. We calculate the similarity of the two trees of Fig. **2** using our Tree Matching algorithm. In Fig. **2**, the distinct tags are <table, tr, td, div, a, p, b> for both the left and the right trees. The first screening procedure shows that the trees are similar. The total number of distinct tags in all levels is 7 for the left tree and 7 for the right tree respectively (1 <table> tag in level 1, 1 <tr> tag in level 2, 1 <td> tag in level 3, 1 tag in level 4, 1 <a> tag and 1 <p> tag in level 5, 1 <b> tag in level 6 of the trees). Therefore, the left tree is retained for further processing as the two trees are similar. The screening procedures will be repeated using the second tree and third tree and so on until the last tree of the group is used if there are more than 2 trees.

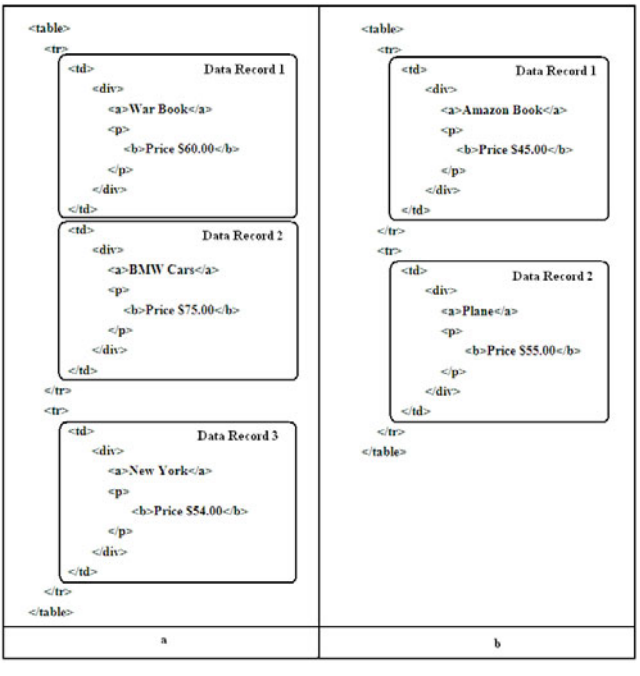

Fig. 2. Two trees with similar structures

Visual Cue Similarity Check

Fig. **3** shows the Lycos search engine results page. As can be seen from Fig. **3**, Data Region 1 (solid rectangles in Fig. **3** are considered not similar because they have bounding boxes with different sizes. Data records in Data Region 2, which are represented by the dotted rectangles in Fig. **3** are similar because they are having bounding boxes of similar sizes. The same applies to Data Regions 3, 4 and 5. SafeMatch will determine the bounding box of each data records and the voting algorithm will mark the data region with one dissimilar data record if the data records differ greatly (say more than 50 pixels) in width and height compared to other data records.

Stage 2: Number of Nodes Filter

Stage 2 of the filtering processes involves removing data regions with less than 3 data records. This Stage is carried out based on Observation 2.

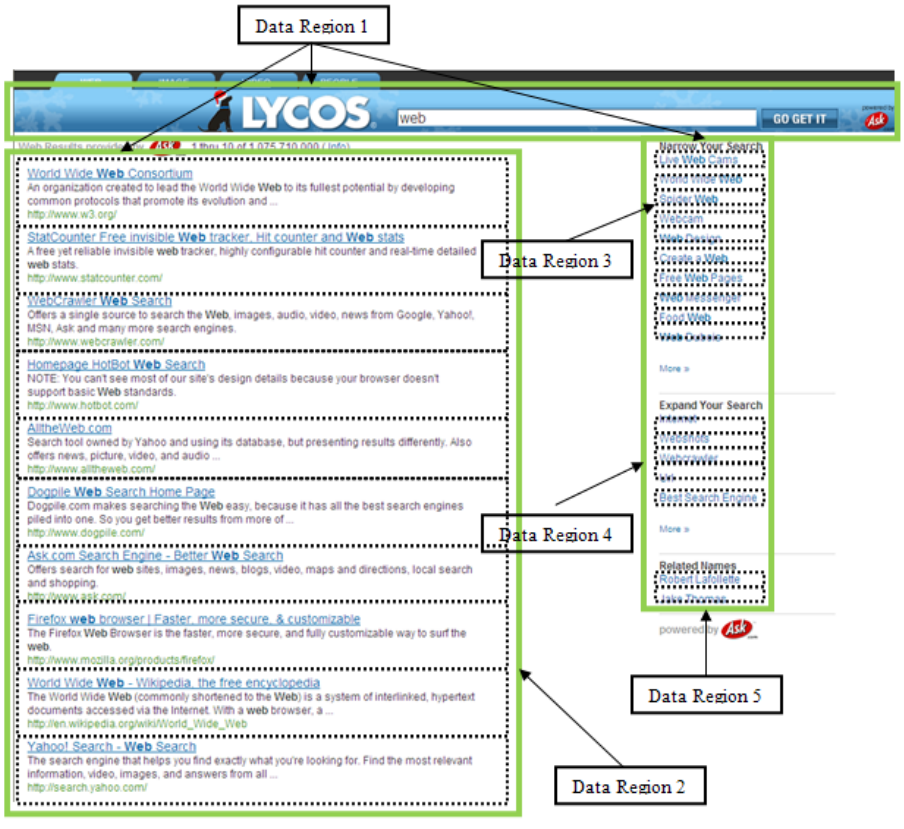

Fig. 3. An example of HTML page containing data regions with similar and dissimilar data records

Stage 3: Largest Scoring Function Filter

After passing through the 3 Stages, SafeMatch will have a list of data regions. It is assumed that these data regions contain data records which are visually and structurally similar as they survived the filtering Stages 1 to 3. Menus which represent the layout of the web page are removed in Stage 2 of the filtering rules. Some of the possible data regions available are the menu bars, advertisements, and data records which are relevant to our work. Each of these data regions is assigned a scoring function. The correct data region is assumed to have the largest score value. Stage 4 is carried out based on Observation 1. We measure the sizes of texts and images of data regions. The sizes of texts and images are measured based on the bounding box of HTML Text and HTML <IMG> tag. Once the area of the bounding box is

ascertained, they are summed up to give a final score value. Each of the data regions has its own score value. It is noted that HTML separator nodes such as
 also contribute to the space occupied in a data record. SafeMatch will therefore measure the size of the bounding box of these nodes.

The scoring function used by SafeMatch wrapper is given below:
a=Size of Texts in a Data Region
b=Size of IMG tags in a Data Region
c=Size of HTML separator nodes in a Data Region
x=Data region

$$Score(x) = a + b + c \ (1)$$

Stage 4: Removing Irrelevant Data from the Relevant Data Region

In this stage, we use the same tree matching algorithm mentioned previously to match all the data records in the relevant data region. We also match data records based on their visual boundaries. Data records that matched each other are put into the list of correct data records. Those data records that are not similar are removed from the relevant data region.

4 Experimental Tests

The dataset used in this study is taken from complete planet repositories (www.completeplanet.com). This dataset contains 250 web pages. The distribution of data for the data set varies, ranging from academic sites, general sites to governmental sites. We compare our work with state of the art wrapper, WISH [4] using this dataset. We do not make comparison with other state of the art wrappers such as ViNT [3] and DEPTA [8] as study in [4] shows that WISH performs comparatively better than ViNT and DEPTA. The measures of wrapper's efficiency are based on three factors, the number of actual data records to be extracted, the number of extracted data records from the test cases, and the number of correct data records extracted from test cases. Based on these three values, precision and recall are calculated according to the formula:

$$Recall=Correct/Actual*100 \ (2)$$
$$Precision=Correct/Extracted*100 \ (2)$$

SafeMatch takes about 400 milliseconds on average to generate a result for a web page and achieves a high recall and precision rate (Table 1). ViNT [3] (state of the art visual assisted wrapper) takes 1200 milliseconds to generate a result page. This shows that the speed of SafeMatch is better than other existing state of the art system (ViNT [3]) while achieving higher accuracy than WISH [4]. SafeMatch outperforms WISH in terms of recall rate and has precision rate comparable to WISH. Our dataset contains mostly complicated web pages, particularly web pages containing a number

Table 1. Experimental result for SafeMatch and WISH

Term	SafeMatch	WISH [4]
Actual	4139	4139
Extracted	3909	3363
Correct	3758	3288
Recall	90.79%	79.43%
Precision	96.13%	97.76%

of data regions which are similar torelevant data region. As SafeMatch uses visual cue that measures text and image size, it will be able to distinguish more efficiently correct data region from incorrect ones. Unlike WISH, SafeMatch is able to match data records with dissimilar tree structures.

5 Conclusions

In this study, we develop a wrapper (SafeMatch) which is able to extract data records from deep webs more efficiently than existing state of the art wrapper WISH. Unlike existing works, our wrapper is able to extract relevant data region which contains search identifiers. We use our voting algorithm in addition to tree matching algorithm and visual cue to extract the relevant data records and remove irrelevant data from the data region accordingly. We also use text and image size to locate and extract the correct data region containing data records. The exact measurements of text and image and the use of visual information to remove dissimilar data records improve the accuracy of our wrapper in data extraction. The accuracy and speed of our wrapper will be useful in meta search engines application.

References

1. Liu, B., Grossman, R., Zhai, Y.: Mining data records in Web pages. ACM SIGKDD, 601–606 (2003)
2. Miao, G., Tatemura, J., Hsiung, W.-P., Sawires, A., Moser, L.E.: Extracting Data Records from the Web Using Tag Path Clustering. ACM WWW, 981–990 (2009)
3. Zhao, H., Meng, W., Wu, Z., Raghavan, V., Yu, C.: Fully automatic wrapper generation for search engines. ACM WWW, 66–75 (2005)
4. Hong, J.L., Siew, E., Egerton, S.: Information Extraction for Search Engines using Fast Heuristic Techniques. DKE 69(2), 169–196 (2010)
5. Simon, K., Lausen, G.: ViPER: augmenting automatic information extraction with visual perceptions. ACM CIKM, 381–388 (2005)
6. Liu, W., Meng, X., Meng, W.: ViDE: A Vision-based Approach for Deep Web Data Extraction. IEEE TKDE 22(3), 447–460 (2009)
7. Su, W., Wang, J., Lochovsky, F.H.: ODE: Ontology-assisted Data Extraction. ACM TODS 34(12) (2009)
8. Zhai, Y., Liu, B.: Web data extraction based on partial tree alignment. ACM WWW, 76–85 (2005)