Automatic Extraction of Logical Web Lists

Pasqua Fabiana Lanotte, Fabio Fumarola, Michelangelo Ceci,
Andrea Scarpino, Michele Damiano Torelli, and Donato Malerba

Dipartimento di Informatica, Universita degli Studi di Bari "Aldo Moro",
via Orabona, 4 - 70125 Bari - Italy
{pasqua.fabiana,ffumarola,ceci,malerba}@di.uniba.it,
{andrea,daniele}@datatoknowledge.it

Abstract. Recently, there has been increased interest in the extraction of structured data from the web (both "Surface" Web and"Hidden" Web). In particular, in this paper we focus on the automatic extraction of Web Lists. Although this task has been studied extensively, existing approaches are based on the assumption that lists are wholly contained in a Web page.They do not consider that many websites span their listing on several Web Pages and show for each of these only a partial *view*. Similar to databases, where a view can represent a subset of the data contained in a table, they split a *logical list* in multiple views (*view lists*). Automatic extraction of *logical lists* is an open problem. To tackle this issue we propose an unsupervised and domain-independent algorithm for *logical list extraction*. Experimental results on real-life and data-intensive Web sites confirm the effectiveness of our approach.

Keywords: Web List Mining, Structured Data Extraction, Logical List.

1 Introduction

A large amount of structured data on the Web exists in several forms, including HTML lists, tables, and back-end Deep Web databases (e.g. Amazon.com, Trulia.com). Caffarella et al. [3] estimated that there are more than one billion HTML tables on the Web containing relational data, and Elmeleegy et al. [5] suggested an equal number from HTML lists. Since the Web is a large and underutilized repository of structured data, extracting structured data from this source has recently received great attention [3,5]. Several solutions have been proposed to find, extract, and integrate structured data, which are used in few public available products like Google Sets and Google Fusion Tables. In addition, the analysis of large amount of structured data on the web has enabled features such as schema auto-complete, synonymy discovery [3], market intelligence [1], question answering [6], and mashup from multiple Web sources [15]. However, only few websites give access to their data through Application Programming Interfaces (e.g. Twitter, Facebook). The majority of them present structured data as HTML and/or backed through "Hidden" Web Database.

In this paper, we focus on the problem of automatic extraction of Web Lists. Several methods have been presented in the literature [13,16,10,9,14] as regards

T. Andreasen et al. (Eds.): ISMIS 2014, LNAI 8502, pp. 365–374, 2014.

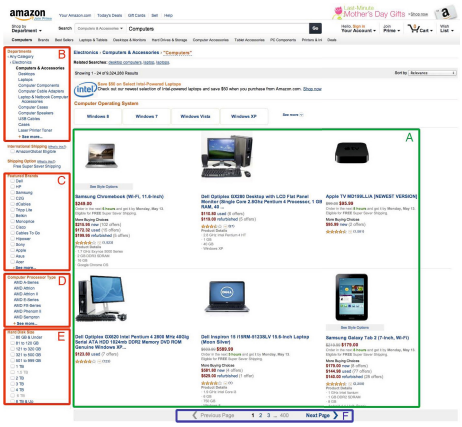

Fig. 1. An example of Amazon Web page

to this task, but they fail to detect lists which span multiple Web pages. This is an open issue, because many web sites, especially data-intensive (e.g. Amazon, Trulia, AbeBooks,...), present their listings as *logical list*, that is, a list spanning multiple pages (*e.g.* computers, books, home listings)[1]. It is as if, each list represents a *view* of the same *logical list*. Similar to databases, where a *view* can represent a subset of the data contained in a table partitioned over a set of attributes, a *logical list* is split in multiple *views* (Web Pages) in order to avoid information overload and to facilitate users' navigation.

For example, Fig. 1 shows a Web page from Amazon.com that contains the results for the query "Computer". On this page, the boxes A, B, C, D, E, F are web lists. The list in the box A shows a *view* of the "Computers" products, that is the top six sorted by relevance, and F allows us to navigate to the other views of the products ordered by relevance. Thus navigating the links in F we can generate the *logical list* of the products for the query "Computer". Boxes B, C, D and E contain respectively the lists representing filters for "Department", "Featured Brands", "Computer Processor Type", and "Hard Disk Size", which are attributes of the *virtual* table "Computer". Moreover, the anchor-text links in boxes B, C, D and E stores valuable information which can be used to annotate data records, and thus to individuate new attributes. For example, the anchor-text links of web list C can be used to index data records based on "Computer brands". Traditionally, search engines use the proximity of terms on a page as a signal of relatedness; in this case the computer brand terms are highly related to some data records, even though they are distant.

Providing automated techniques for *logical list* extraction would be a significant advantage for data extraction and indexing services. Existing data record

[1] The motivations behind this approach are as well technical (reducing bandwidth and latency), and non technical as avoiding information overload or maximizing page views.

extraction methods [13,16,9,14] focus only in extracting *view* lists, while several commercial solutions[2] provide hand-coded rules to extract *logical lists*.

In this paper, we face this issue by proposing a novel unsupervised algorithm for automatic discovery and extraction of *logical lists* from the Web. Our method requires only one page containing a *view list*, and it is able to automatically extract the *logical list* containing the example *view list*. Moreover, during the process, it enriches the list's elements with the pair <*url, anchor-text*> used for the extraction task. We have validated our method on a several real websites, obtaining high effectiveness.

2 Definitions and Problem Formulation

In this section, we introduce a set of definitions we will use through the paper.

Definition 1. The Web Page Rendering *is the process of laying out a spatial position of all the text/images and other elements in a Web Page to be rendered.*

When an HTML document is rendered in a Web browser, the CSS2 visual formatting model [11] represents the elements of the document by rectangular boxes that are laid out one after the other or nested inside each other. By associating the document with a coordinate system whose origin is at the top-left corner, the spatial position of each text/image element on the Web page is fully determined by both the coordinates *(x,y)* of the top-left corner of its corresponding box, and the box's height and width.

Property 1. The spatial positions of all text/image elements in a Web Page define the *Web Page Layout*.

Property 2. As defined in [7], each *Web Page Layout* has a tree structure, called *Rendered Box Tree*, which reflects the hierarchical organization of HTML tags in the Web page.

As we can see in Fig. 2, on the left there is the Web Page Layout of the Web page in Fig. 1. On the right, there is its Rendered Box Tree. The technical details of building Rendered Box Trees and their properties can be found in [7]. Under the Web page layout model, and the Rendered Box Tree we can give the definition for Web lists.

Definition 2. ***Web List****: It is collection of two or more objects, under the same parent box and visually adjacent and aligned on a rendered web page. This alignment can occur via the x-axis (i.e. a vertical list), the y-axis (i.e. horizontal list), or in a tiled manner (i.e., aligned vertically and horizontally) [8].*

For example the list A in Fig. 1 is a tiled list, while B is a vertical list and F is a horizontal list.

[2] Lixto, Screen Scraper Studio, Mozenda Screen Scaper.

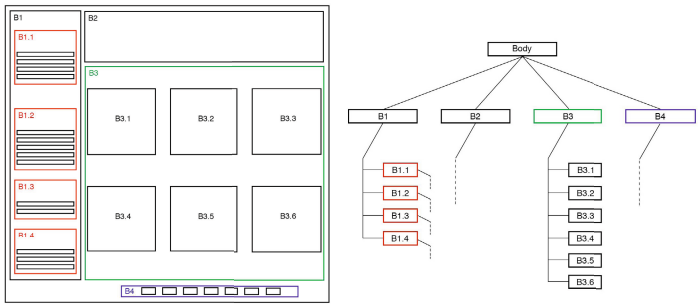

Fig. 2. An example of Rendered Box Tree

The list's elements can be called as *Data Records*. Similar to the concept of data records into database, data records into a web page are a set of similar and structured objects containing information. Typically, they are formatted using similar HTML tags (i.e. the same HTML structure).

Definition 3. ***Logical List****: It is a list whose Data Records are distributed on more then one Web Pages.*

An example is shown in Fig. 3, where the boxes A1 and A2 represent a part of a *logical list*.

Definition 4. ***View List****: It is a view of a logical list, whose Data Records are all contained in same Web page.*

For Example the list F in Fig. 1 is an example of a view list. In fact, it contains only some of data records belonging to its logical list (that is the *pagination list*).

Definition 5. ***Dominant List:*** *It is the view list of interest, containing data records from the logical list that we want to extract.*

For example the list A in Fig. 1 is the Dominant List for the given Web page.

3 Methodology

In this section we describe the methodology used for *logical list* extraction. The algorithm employs a three-step strategy. Let P a Web Page, it first extracts the set L^P of the lists contained in P; in the second step, it identifies the *dominant list* $l^P_{dom} \in L$; finally, it uses l^P_{dom} to discover the *logical list* LL which includes l^P_{dom} as sub-list. These steps are detailed in the following sub-sections.

3.1 List Extraction

Given a Web Page P as input, its lists $L = \{l_1, l_2, \ldots, l_n\}$ are extracted through running an improved version of HyLiEn [8]. With respect to HyLiEn, we made

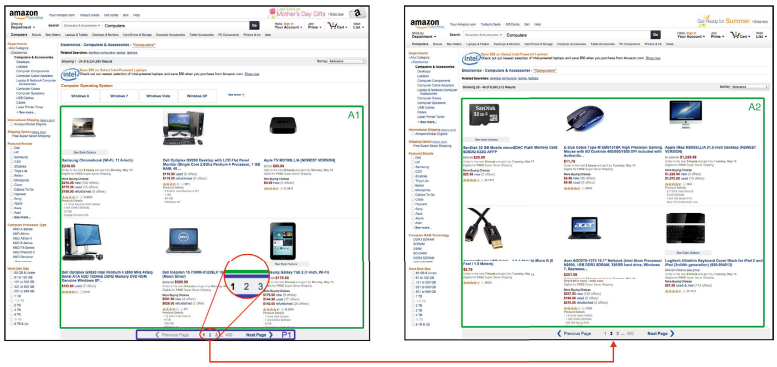

Fig. 3. An example of *logical list* for Amazon's products

several improvements. First, to render the Web pages we removed the dependency to the open source library *CSSBox*[3], because we found that this library was not able to correctly render several Web pages. We implemented a WebKit wrapper, called *WebPageTraverser*[4], which is released as open source project. Given as input the url of a Web page P, *WebPageTraverser* outputs a JSON[5] representation of the P using the *rendered box tree model*. Second, to compute the similarity of two sub-trees in *rendered box tree* (see Prop. 2) we adopted the HTML tag distance, presented in [2] instead of the string edit distance used by HyLiEn. Although, our current implementation uses the HyLiEn algorithm to obtain Web lists our solution is independent of any specific list-extraction algorithm. We used HyLiEn because it showed interesting result compared to the state of art algorithms for List Extraction [8].

3.2 Dominant List Identification

Given a Web Page P and the set of list $L = \{l_1, l_2, \ldots, l_n\}$ extracted in the first step, we use three measures to identify the *dominant list* of P:

- **Centrality.** Given a list $l_i \in L$, the *centrality* of l_i w.r.t P is obtained by computing the Euclidean distance between the center of the parent-box of l_i and the center of root-box of P.
- **Area Ratio.** Given a list $l_i \in L$, the *area ratio* of l_i w.r.t P is the size of the box containing l_i divided the size of root-box of P.
- **Text-Tag Ratio.** Given a list $l_i \in L$, and let m the length of l_i, the *text-tag ratio* of l_i is computed as:

$$\frac{1}{m}\sum_{j=0}^{m}\frac{chars(l_i[j])}{tag(l_i[j])} \tag{1}$$

[3] `http://cssbox.sourceforge.net`
[4] `https://bitbucket.org/wheretolive/webpagetraverser`
[5] `http://www.json.org/`

where $tag(l_i[j])$ is the number of HTML tags contained in the j-th data record of l_i and $chars(l_i[j])$ is the total number of characters contained in $l_i[j]$. Before that the text-tag ratio is computed, *script* and *remark* tags are removed because this information should be not considered in the count of non-tag text.

In particular the *Dominant list* of P is the list with the highest sum of contributions:

$$\arg\max_{l_i \in L} \frac{\alpha_1}{centrality(l_i)} + \alpha_2 areaRatio(l_i) + \alpha_3 textTagRatio(l_i) \qquad (2)$$

where $centrality(l_i), areaRatio(l_i)$ and $textTagRatio(l_i)$ are respectively the centrality measure, area ratio and text-tag ratio of a list l_i contained in L. $\alpha_1 = \alpha_2 = \alpha_3$ are set to 0.3 to give the same weight to each measure.

Algorithm 1. `LogicalListDiscovery`

input : dominant list l^P_{dom}, set $L_- = \{L \setminus l^P_{dom}\}$
output: logical list LL

```
1 LL = {l^P_dom};
2 forall the l ∈ L_− do
3     forall the u ∈ l do
4         L_u ← HyLiEn(u);
5         L_u.filterSimilarity(l^P_dom, α);
6         LL.add(L_u);
7 LL ← LL.flatMap();
8 LL ← LL.removeDuplicates();
9 return LL;
```

3.3 Logical List Discovery

Identified the dominant list l^P_{dom} of the Web Page P, the last step of the algorithm is to discover the logical list LL containing l^P_{dom}. This is done by taking advantage of the regularities of Web Sites. As described by Crescenzi et al. [4], Web page links reflect the regularity of the web page structure. In other words, links that are grouped in collections with a uniform layout and presentation usually lead to similar pages. Link-based approaches are used in the literature for tasks strictly related to the one solved by our method. For instance, Lin et al. [12] used Web links to discover new attributes for web tables by exploring hyperlinks inside web tables. Lerman et al. [10] uses out-links to "detail web pages" in order to segment Web tables. In this paper, we successfully use links grouped as lists to navigate Web pages and to discover *logical lists*.

The algorithm 1 describes the approach used. It takes as input the dominant list l^P_{dom}, the minimum similarity threshold α, and the set of the lists L extracted from P. It iterates over all the lists in the set $L_- = \{L \setminus l^P_{dom}\}$ (line 1), and, for each url u in l_i it alternates, (i) the extraction of the set list L_u contained in the Web Page U having u as url (line 4) to, (ii) the filtering of all the lists in L_u which have a similarity with l^P_{dom} lower than α (line 6). At each iteration, all the lists resulting from step (ii) are added to LL (line 7). Finally, LL is flattened and all the duplicate elements are merged (lines 8-9). Moreover, during the process all the anchor text of url u are used as attributes to annotate the discovered *view* lists and are reported in the final *logical list* LL.

4 Experiments

In this section we presents the empirical evaluation of the proposed algorithm. We manually generated and verified a test dataset. In particular, for the experiment, we select 40 websites in different application domains (music shops, web journals, movies information, home listings, computer accessories, etc.) with list elements presented in different ways. For the deep-web databases, we performed a query for each of them and collected the first page of the results list, and for others we manually select a Web page. Table 1 shows in the first column the ground truth, that is, the number of data records which belong to the *logical list* to be extracted. The dataset is composed of 66.061 list elements extracted from 4405 Web pages. We rendered each Web page and we manually identified (i) *dominant list* and, (ii) following the out-links of the other lists in the pages we annotated *logical lists*. This task required around 7 days of 4 people.

To the best of our knowledge the task of *Logical List Discovery* is novel, and there are not any other methods to compare with. So, we evaluated the effectiveness of our algorithm by using *precision*, *recall* and *f-measure* metrics, computed over the number of *logical list* elements to be extracted (manually verified) w.r.t to the algorithm results. In particular, the precision is the measure of, how many of the extracted *view* lists belong to a *logical list*. The recall allows us to measure how many of the discovered *view* lists are true positive element of a *logical list*. We also included the *F-Measure* which is the weighted harmonic means of *precision* and *recall*. These metrics are evaluated counting how many data records of the *logical list* are found in the *view* lists.

$$precision = \frac{TP}{TP + FP},\ recall = \frac{TP}{TP + FN}, F - measure = \frac{2(precision \times recall)}{precision + recall} \tag{3}$$

4.1 Results

The execution of the algorithm requires two parameters which are empirically set to $\alpha = 0.6$ and $\beta = 50$. These parameters are need by the HyLiEn Algorithm. Our methods uses α during the *Logical List Discovery* step.

Table 1 presents the main results. The first column holds for each logical list the number of data records to extract. The second and the third columns contain the number of true positive and the number of false negatives data records. We do not plot the number of false positives, because our algorithm outputted always 0 false positives during the experiment evaluation. Finally, the fourth, fifth and sixth columns show the values for precision, recall and f-measure.

In general, the experimental results show that our algorithm is able to discover *logical lists* in a varying set of Web sites (that is, it is not domain dependent). Moreover, the quality of the results are not correlated to how the lists are rendered in Web pages (i.e. horizontal, vertical and tiled). In average, it achieves 100% for Precision, 95% for Recall and a F-Measure 97%. With respect to the ground truth, the algorithm does not extract any False Positive, and it outputs only 466 False Negatives. In general, it returns perfect results (100% precision and recall) for several kind of websites spanning different applications domain, but there are some of them which presents values for recall ranging from 81% and 91%. Considering "last.fm", which gave a recall equal to 81%, we found that the presentation of the data records is sometime quite different, because of the high variance in the number of the "similar to" tags (which are presented as HTML <a>) assigned to each listing. Analyzing other examples such as "IlSole24Ore.it" and "RealEstateSource.au" we found the same problem, that is, the presentation of the data records is quite variable across the Web pages, and so the HyLiEn algorithm sometimes misses some of the data records. Anyway we see that the proposed algorithms is effective is able to discover *logical lists* on different type of websites.

5 Conclusions and Future Works

In this paper, we have presented a new method for *Logical List Extraction*. Our method solves the open issue of discover and extract lists which spans multiple Web pages. These *logical lists* are quite common in many websites, especially data-intensive, where their listings are split on multiples pages in order to avoid information overload and to facilitate users' navigation. However, the data stored in such *logical list* need to be automatically extracted to enable building services for market intelligence, synonyms discovery, question answering and data mashup. Experimental results show that our new method is extremely accurate and it is able to extract *logical lists* in a wide range of domains and websites with high precision and recall. Part of this future work will involve tasks such as indexing the Web based on lists and tables, answering queries from lists, and entity discovery and disambiguation using lists.

Acknowledgements. This work fulfills the research objectives of the PON 02 00563 3470993 project "VINCENTE - A Virtual collective INtelligenCe ENvironment to develop sustainable Technology Entrepreneurship ecosystems" funded by the Italian Ministry of University and Research (MIUR).

Table 1. Discovered *logical list* elements for Web sites dataset

Website	*Ground*	*TP*	*FN*	*Precision*	*Recall*	*F-measure*
BariToday.it	904	904	0	100%	100%	100%
Subito.it	1000	1000	0	100%	100%	100%
GitHub.com	100	100	0	100%	100%	100%
TestoLegge.it	360	360	0	100%	100%	100%
Zoopla.co.uk	597	597	0	100%	100%	100%
FindAProperty.co.uk	60	60	0	100%	100%	100%
Savills.co.uk	232	232	0	100%	100%	100%
AutoTrader.co.uk	60	60	0	100%	100%	100%
EbayMotors.com	3925	3925	0	100%	100%	100%
Doogal.co.uk	38240	38240	0	100%	100%	100%
RealEstateSource.com	368	316	62	100%	85%	91%
AutoWeb.co.uk	180	180	0	100%	100%	100%
TechCrunch.com	434	422	12	100%	95%	98%
Landsend.com	1243	1243	0	100%	100%	100%
TMZ.com	300	300	0	100%	100%	100%
IlSole24Ore.it	510	445	65	100%	81%	86%
GoBari.it	350	340	10	100%	97%	98%
AGI.it	60	60	0	100%	100%	100%
BBCNews.co.uk	347	310	37	100%	89%	94%
milano.corriere.it	30	30	0	100%	100%	100%
torino.repubblica.it	70	68	2	100%	98%	99%
Ansa.it	1506	1479	27	100%	98%	99%
LeMonde.fr	445	418	27	100%	94%	97%
Time.com	377	377	0	100%	100%	100%
aur.ArchLinux.org	575	575	0	100%	100%	100%
Immobiliare.it	609	536	73	100%	86%	93%
bitbucket.org	130	130	0	100%	100%	100%
MyMovies.com	563	515	48	100%	92%	96%
Trulia.com	3300	3300	0	100%	100%	100%
YouTube.com	580	567	13	100%	98%	99%
FileStube.com	332	304	28	100%	91%	95%
Last.fm	60	41	19	100%	68%	81%
Bing.com	130	130	0	100%	100%	100%
addons.mozilla.org	984	939	45	100%	95%	97%
AutoScout24.com	840	840	0	100%	100%	100%
Facebook.com	2820	2820	0	100%	100%	100%
SlideShare.net	2037	2037	0	100%	100%	100%
Gazzetta.it	970	970	0	100%	100%	100%
ElPais.es	294	285	9	100%	98%	99%
StackOverflow	585	585	0	100%	100%	100%
Sums and Averages	66.527	66.061	466	100%	95%	97%

References

1. Baumgartner, R.: Datalog-related aspects in lixto visual developer. In: de Moor, O., Gottlob, G., Furche, T., Sellers, A. (eds.) Datalog 2010. LNCS, vol. 6702, pp. 145–160. Springer, Heidelberg (2011)
2. Bing, L., Lam, W., Gu, Y.: Towards a unified solution: Data record region detection and segmentation. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM 2011, pp. 1265–1274. ACM, New York (2011)
3. Cafarella, M.J., Halevy, A., Madhavan, J.: Structured data on the web. Commun. ACM 54(2), 72–79 (2011)
4. Crescenzi, V., Merialdo, P., Missier, P.: Clustering web pages based on their structure. Data Knowl. Eng. 54(3), 279–299 (2005)
5. Elmeleegy, H., Madhavan, J., Halevy, A.: Harvesting relational tables from lists on the web. The VLDB Journal 20(2), 209–226 (2011)
6. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, pp. 1535–1545. Association for Computational Linguistics, Stroudsburg (2011)
7. Fumarola, F., Weninger, T., Barber, R., Malerba, D., Han, J.: Extracting general lists from web documents: A hybrid approach. In: Mehrotra, K.G., Mohan, C.K., Oh, J.C., Varshney, P.K., Ali, M. (eds.) IEA/AIE 2011, Part I. LNCS, vol. 6703, pp. 285–294. Springer, Heidelberg (2011)
8. Fumarola, F., Weninger, T., Barber, R., Malerba, D., Han, J.: Hylien: A hybrid approach to general list extraction on the web. In: Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M.P., Bertino, E., Kumar, R. (eds.) WWW (Companion Volume), pp. 35–36. ACM (2011)
9. Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., Pollak, B.: Towards domain-independent information extraction from web tables. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 71–80. ACM, New York (2007)
10. Lerman, K., Getoor, L., Minton, S., Knoblock, C.: Using the structure of web sites for automatic segmentation of tables. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, SIGMOD 2004, pp. 119–130. ACM, New York (2004)
11. Lie, H.W., Bos, B.: Cascading Style Sheets: Designing for the Web, 3rd edn., p. 5. Addison-Wesley Professional (2005)
12. Lin, C.X., Zhao, B., Weninger, T., Han, J., Liu, B.: Entity relation discovery from web tables and links. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, pp. 1145–1146. ACM, New York (2010)
13. Liu, B., Grossman, R.L., Zhai, Y.: Mining web pages for data records. IEEE Intelligent Systems 19(6), 49–55 (2004)
14. Liu, W., Meng, X., Meng, W.: Vide: A vision-based approach for deep web data extraction. IEEE Transactions on Knowledge and Data Engineering 22(3), 447–460 (2010)
15. Maximilien, E.M., Ranabahu, A.: The programmableweb: Agile, social, and grassroot computing. In: Proceedings of the International Conference on Semantic Computing, ICSC 2007, pp. 477–481. IEEE Computer Society, Washington, DC (2007)
16. Miao, G., Tatemura, J., Hsiung, W.: Extracting data records from the web using tag path clustering. In: The World Wide Web Conference, pp. 981–990 (2009)