

Taxonomy Based Data Extraction from Multi-item Web Pages

Hui Guo and Amanda Stent

Stony Brook University, Stony Brook, NY 11794-4400
{huiguo1978, amanda.stent}@gmail.com

Abstract. In this paper, we present a taxonomy-driven approach to the extraction of data records from web pages containing multiple similar items. In our approach, we first automatically extract a taxonomy from the web for the target domain, then extract data records from web pages in the target domain, and finally use the automatically-extracted taxonomy to automatically annotate feature values in the text of each data record. A key advantage of our approach is that it permits data extraction without human labeling or training. Our evaluation results show that our approach performs well for this task.

1 INTRODUCTION

The task of web data extraction is to extract content from web pages for other applications (e.g. summarization, task learning). However, web pages are heterogeneous; many pages are semi-structured and noisy. To extract information from these pages, several challenges must be overcome. First, extraneous information (e.g. navigation links, ads) must be removed (**noise removal**). Second, the page must be segmented into data records (**record extraction**). Finally, the fields in each record must be labeled (**field labeling**).

In this paper, we describe an approach to data extraction from multi-item web pages that: is *unsupervised*; can *automatically segment data records* in web pages; and can *automatically label fields* in those records. Our approach uses a novel method for automatic extraction of a domain-specific taxonomy from structured portions of web pages, also described in this paper. We have evaluated our approach on commercial web pages and demonstrated that it is feasible. Our data, extracted taxonomies and detailed results for all experiments can be found at <http://www.cs.sunysb.edu/~nlp/webmining/>.

2 Related Work

Noise removal An important problem for information extraction from the web is the identification and removal of noise (e.g. web site navigation links and ads). [1] proposes some techniques to eliminate noise from web pages based on the following observation: in a given web site, noisy blocks usually have similar content and presentation styles, while the main content blocks are often diverse in

their actual content and/or presentation styles. However, this observation about presentation styles is not always true. For example, the product pages from Buy.com contain ads with diverse presentation styles, while the main content is presented using lists only. Our method for detecting noise is described in Section 3.1.

Record extraction Several researchers have done work on wrapper induction. A wrapper is a program that extracts data from web pages and puts it in a database. Some wrapper induction methods require a set of manually labeled training examples ([2–5]). Manual labeling is time consuming and costly, only worthwhile if the wrapper will be used repeatedly. Our approach does not use manual labeling.

Work on totally or mostly automatic extraction from web pages includes [6–10]. IEPA [6] extracts only tables and lists from web pages. Both RoadRunner [7] and EXALG [8] extract data from web pages by comparing web pages from the same site (in the same class); the former uses heuristics for data extraction, while the latter searches for patterns that cross pages. In these systems, the web page is treated as a simple sequence of tokens.

Other researchers use the DOM tree from a web page. [9] extract data by analysis of the DOM tree, but do not generate field labels for the extracted data. By contrast, the approach described in [10] is able to extract data records (using the DOM tree) and semantically annotate the extracted data (using hand-labeled concepts). Both of these approaches treat all nodes of the DOM tree equally, even though some node types are more similar than others. By contrast, we use a measure of node type similarity in our approach (see Section 3.2).

Field labeling Most existing web extraction systems cannot assign field labels to extracted data records. Here, we discuss those that can. DeLa [11] uses HTML forms to automatically label fields in data records; this approach only works for web-based query interfaces to databases. By contrast, [10, 12–15] use human-created resources for field labeling. The approaches described in [10, 12, 13] use human-labeled training data or human-created text-matching patterns, while [14, 15] use ontologies. [14] use a human-created ontology to automatically generate a database schema and to generate rules for matching constants and keywords. They then use the matching rules to identify potential feature values and feature label keywords in web pages. Most recently, [15] use declarative instance semantics added to OWL and human-built ontologies to label fields in data extracted from web pages. In our approach we use a taxonomy; our taxonomy is automatically acquired and includes automatically acquired text matching patterns for field labeling. As part of our evaluation of our approach, we compare our results to the output of the demo system for [15].

3 Our Approach

Our approach has two stages. In the first, **taxonomy extraction**, we automatically extract a domain-specific taxonomy. For this stage, we use web pages containing only one item per page, with a *feature table* (a list of features and values for that item) on each page. In the second stage, **data extraction**, we use the taxonomy to automatically extract data records and label fields. For this stage, we use web pages containing multiple items per page (as in Fig. 1).



Fig. 1. A multi-item page from Staples.com.

Both stages of our approach use information from the DOM trees of web pages. DOM trees contain type and structural information for nodes and their content (e.g. images, text). Some nodes are visually similar (e.g., plain text, bold text, text with links), while others are visually different (e.g., a picture and a table). Also, some nodes contain more extractable information than other nodes (e.g., different amounts of text). Based on these observations, we defined two measures for nodes in the DOM tree. The *tag similarity* between any two DOM tree nodes is defined using a table like Table 1. The *node weight* of a single node is computed using rules like those in Table 2. These measures are determined heuristically rather than using machine learning because for this task machine learning leads to overfitting, while our heuristics work with high accuracy and are quick. Using these two measures, we can compute the *structural similarity* between two DOM trees (see Fig. 2).

Table 1. Similarity Degree Matrix Between Tags

	Text	Link	...	Image
Text	1	0.9	...	0.1
Link	0.9	1	...	0.1
...
Image	0.1	0.1	...	1

Table 2. Rules for Node Weights

Type of Node	Rule
Plain Text	The number of words in the text
Bold Text	The weight of the text times a factor
Image	The number of pixels times a factor
...	...

Algorithm: Structure_Similarity_Degree(A, B)
input: A, B are DOM tree nodes

```

rootSimDegree := Tag_Similarity_Degree(root(A), root(B))
if rootSimDegree is less than a threshold
then return 0;
else
  m:= the number of first-level sub-trees of A;
  n:= the number of first-level sub-trees of B;
  Initialization: M[i, 0]:= 0 for i = 0, , m;
  M[0, j] := 0 for j = 0, , n;
  for i = 1 to m do
  for j = 1 to n do
    M[i,j]:=max(M[i,j-1], M[i-1, j], M[i-1, j-1]+W[i, j]);
    where W[i,j] = Structure_Similarity_Degree(Ai, Bj)
      * (Weight(Ai) + Weight(Bj))
  Total_Weight = Weight(A) + Weight(B)
return (M[m, n]+ rootSimDegree*rootWeight)/Total_Weight

```

Fig. 2. Algorithm for computing structural similarity

3.1 Taxonomy Extraction

This stage is a pipeline of four processes: page segmentation, noise removal, feature table detection, and taxonomy creation.

Page Segmentation The task in this stage is to segment the web page using the visual cues provided by the HTML markup. Some markup and content in web pages is typically invisible (e.g. headers); we call these items *invisible content*. Some items, while visible, serve only to separate other parts (e.g. the <HR> tag); we call these items *separators*. We call other items in the web pages that are not invisible and are not separators *elements*. Elements represent the viewer's idea of web page organization. We call groups of items *blocks*. A block can contain presentation elements, separators, invisible content and other blocks.

As the above definitions make clear, if we can identify separators, we can identify elements and group them into blocks. So we just need an efficient separator detection algorithm. However, a formatting element (e.g. an image of a horizontal line) may be a separator at one position in a page but not another.

[16] present an approach to separator detection in HTML based on computing the visual difference degree between neighboring blocks. We apply modified

versions of their rules to compute the visual difference degree of a tag that may be a separator. If the visual difference degree exceeds a preset threshold, we mark it as a separator and mark the structures on each side as blocks.

To perform page segmentation, we apply our separator detection algorithm to the DOM tree nodes in a top-down fashion. Invisible nodes are ignored. Elements between separators are put into blocks. The output from our page segmentation process is a *partition tree* in which each node is either a block or a set of blocks.

Noise Removal Noise in web pages includes advertisements, navigation links and page borders. Our approach to noise removal is a modification of that described in [1]. Their definition of noise is based on two heuristics: the importance of a node is proportional to the diversity of presentation styles used in it; and its importance is proportional to the diversity of its content. For our data, the first heuristic is not generally true, so we only use the second.

To perform noise removal, we compare the partition trees of pairs of web pages from the same site. Starting from the root of each partition tree, we check whether the pairs of blocks are content similar. Two blocks are *content similar* if and only if they are structure similar and their corresponding leaves contain similar content. If the blocks are content similar, we assume they are noise and remove them; otherwise, we recurse on their children. The output from this process is a slimmed-down partition tree.

Feature Table Detection Many web pages contain feature tables similar to the one in Fig. 3. These feature tables are frequently not represented as HTML tables in the web page (and not all HTML tables are feature tables). However, feature table rows are structurally similar: each row in a feature table contains the same number of columns, and for pages from the same web site, the feature name lists are similar. We use these observations to find feature tables in the partition tree. Starting from the leaves of the partition tree, we work up until we reach a block node. Then, if the block’s children consist of sequences of equal length that are structurally similar (see Fig. 2), we label that block as a feature table. We continue to the root of the partition tree.

Taxonomy Creation We process the feature name-value pairs in each feature table and build a taxonomy from them. The taxonomy consists of a hierarchy of *frames* (one frame per object in the taxonomy). A frame consists of a set of *fields*, one per feature in the feature table. Each field has a feature name and a set of text-matching *terms*, with their relative frequencies. The terms are unigrams and bigrams extracted from the feature value’s text in the feature table (after removing non-ASCII characters and stop words). The relative frequency for each term is computed by dividing the frequency of occurrence of the term in the current field by the overall frequency of the term in our data. Each feature table contributes to at least one frame: if multiple feature names in a feature table have the same starting term (e.g. “Processor”), we merge them into their own frame.

Product Details	
Warranty Terms - Parts	1 year limited (6 months for battery)
Warranty Terms - Labour	1 year limited
Product Height	1.4"
Product Width	10.5"
Product Weight	3.6 lbs.
Product Length	8"
Processor Brand	Intel® Celeron® M ULV
Processor Speed	1.0GHz
Display Type	WXGA widescreen with AveraBrite technology (1280 x 768)
Screen Size	10.6"
System Bus	400MHz

Fig. 3. The feature table from Bestbuy.com.

We may extract a taxonomy from pages from a single site or from multiple sites. If multiple sites are included, we extract a taxonomy from each site and then merge them. Different sites may use different names for the same feature (e.g. “System Memory” and “RAM”). We currently make no attempt to merge these, but simply leave them as separate fields in the merged taxonomy.

3.2 Data Extraction

This stage, which operates on pages containing multiple data records, is a pipeline of four processes: page segmentation, noise removal, record extraction, and field labeling. The first two have already been described above.

Record extraction We use a modification of the approach described in [9]. First, we segment the page and perform noise removal. Then, we take the block in the partition tree with the greatest node weight, on the assumption that this is where the main content in the page will be. We find blocks in this block that are structurally similar. Each of these blocks is assumed to be a data record.

Field labeling In this stage, we use the automatically extracted taxonomy to label text in the data record and so populate one or more taxonomy frame instances. First, we segment the text as follows: each occurrence of an HTML tag is a text separator; each occurrence of punctuation (e.g. “,”, “;”, “.”) is a text separator; and coordinating and subordinating conjunctions (e.g. “and”, “or”, “but”) are text separators. For example, the text “*Powered by the Intel Pentium M processor, fast 533MHz FSB, DDR2 memory and Intel Wireless card*”, is split into the four strings “*Powered by the Intel Pentium M processor*”, “*fast 533MHz FSB*”, “*DDR2 memory*” and “*Intel Wireless card*”.

Second, we find text segments that could be values for fields in the taxonomy. Each field has a feature name and a set of terms with associated relative frequencies (RFs) from the training data. We define a feature name factor (FNF) for the similarity between a feature name and a text segment: $FNF = 1 + N_i/N_t$,

where N_i is the number of words in both the feature name and the text segment and N_t is the total number of words in the feature name. To approximate the match between a text segment and a taxonomy field we use the sum of RFs for all terms and the FNF for the feature name: $FeatureMeasure_{field} = \sum_{i=1}^{\#terms} RF_{terms[i]} * FNF_{feature\ name}$. This measure gives greater weight to the field’s feature name, which is a stronger indicator.

We compute the *FeatureMeasure* for each field for each text segment. We assign to the text segment the field with the highest *FeatureMeasure*, unless all of them are lower than a threshold, in which case we discard the text segment. Finally, we construct a two-column *frame table* for each data record. The first column contains all the fields with at least one matching text segment. The second contains all text segments matching that field.

4 EVALUATION

In this section, we present two experiments. First, we evaluate our taxonomy extraction algorithm. Second, we compare the performance of our system to that of the demo system for the approach described in [15].

4.1 Taxonomy extraction

In our approach, there are multiple ways to acquire taxonomies. Here, we compare three in the computer purchasing domain:

- **Single** Learn a taxonomy from a single site; label all pages.
- **Separate** Learn taxonomies from multiple sites; label each page using its site-specific taxonomy.
- **Merged** Learn taxonomies from multiple sites, merge them into one taxonomy and label all pages.

For **single**, we used the best taxonomy we were able to extract, that from Bestbuy.com, and evaluated on ten websites. For **separate**, we extracted taxonomies from pages from Staples.com, Compusa.com and Bestbuy.com and evaluated each on pages from the same site. For **merged**, we merged the three taxonomies from **separate** as described in Section 3.1, and evaluated on the same ten websites used for **single**.

For each method, we asked two non-expert human evaluators to evaluate the frame table for each data record, comparing it to the original web page. For each field, the evaluator was instructed to label the extracted text segment(s) as *correct (C)*, *partially correct (P)*, *incorrect (I)*, or *extraneous (E)*. Finally, the evaluator was then asked to identify all *missing (M)* (not extracted) information in the data record and any missing data records (there were no missing data records for any method). From the evaluation results, we computed the number of *correct (C)*, *partially correct (P)*, *incorrect (I)*, *spurious (S)* and *missing (M)* text segments for each website. We used these to compute recall and precision for each method: $recall = \frac{C+P*0.5}{possible}$, $precision = \frac{C+P*0.5}{actual}$, $F = \frac{2*recall*precision}{recall+precision}$

Table 3. Comparison of different approaches to taxonomy acquisition in the computer domain

Method	POS	ACT	COR	PAR	INC	MIS	SPU	REC	PRE	F
single	844	646	586	12	0	246	48	0.7	0.91	0.79
separate	275	233	197	4	0	74	32	0.72	0.85	0.77
merged	913	860	734	37	15	127	74	0.82	0.87	0.84

where: $possible = C + P + I + M$ and $actual = C + P + I + E$. Our overall evaluation results for each method are shown in Table 3. The best-performing method is **merged**. Surprisingly, **single** performs better than **separate**. This indicates that if a fairly good taxonomy can be extracted from even one site, our approach may give good performance. Table 4 shows that we can achieve similar performance in other domains.

Table 4. Cross-domain performance of our algorithm using merged taxonomies, ten sites per domain, average of 11 products per site

Domain	POS	ACT	COR	PAR	INC	MIS	SPU	REC	PRE	F	Best site	Worst site
Computer	913	860	734	37	15	127	74	0.82	0.87	0.84	Amazon	Bestbuy
Camera	530	484	388	16	8	118	72	0.74	0.81	0.77	Compusa	Amazon
Microwave	492	407	339	50	3	100	15	0.73	0.89	0.8	Outpost	Amazon

A major cause of errors in our evaluation is key words missing from the taxonomy. For example, in our taxonomy the field “Optical Drive” has the terms “DVD+RW”, “DVD-RW” and “DVD+-RW”, but doesn’t have the term “DVD+/-RW”. So our system didn’t recognize “DVD+/-RW” as “Optical Drive”. Another cause of errors is that text in digital camera pages contains relatively little punctuation and few stop words, leading to text segmentation failures. For example, “Canon X300 8MP Camera” is not split into two text segments, so we cannot simultaneously identify the camera’s resolution and manufacturer.

4.2 Comparative evaluation

In the absence of a shared task or standard evaluation set, it is difficult to compare different approaches to the problem of data extraction from web pages. However, [15] have made a demo system and sample web pages available at <http://www.deg.byu.edu/>. We used their pages for three domains, running them through both their demo system and our system and using the human evaluation method described above. The results, shown in Table 5, highlight the advantages and disadvantages of our approach. We achieved higher recall in the digital camera domain because our automatically extracted taxonomy includes many fields missing from their hand-created ontology (e.g. LCD, memory type). Our performance was lower in the other two domains because the data records include

text that is hard to segment. For example, our system could not segment the text “ViewSonic Optiquest Q51 15”/13.7” color 1024@67Hz .28mm OSD FS PnP”. Overall, the two systems perform similarly, even though their system relies on extensive hand-coding of ontologies and text-matching patterns, while ours uses an automatically acquired taxonomy and automatically-acquired term sets.

We were not able to compare the performance of the demo system for [15] to ours on our data because their system cannot parse our web pages. Comparing their ontologies to our taxonomies, we can see that they have less information per domain (for example, far fewer makers of digital cameras), so we hypothesize that our system would outperform theirs on our data. However, they also have web pages from other domains that we could not use, because they don’t include single-item pages with feature tables from which we can extract a taxonomy and multi-item pages for data extraction and feature labeling.

Table 5. Comparison with BYU system

Domain	System	POS	ACT	COR	PAR	INC	MIS	SPU	REC	PRE	F
Digital Cameras	Our system	80	62	51	4	1	24	6	0.66	0.85	0.75
	BYU system	80	30	29	0	1	50	0	0.36	0.97	0.53
Cars	Our system	57	46	44	0	1	12	1	0.77	0.96	0.85
	BYU system	57	50	50	0	0	7	0	0.88	1	0.93
Monitors	Our system	309	207	182	1	3	123	1	0.66	0.98	0.78
	BYU system	309	235	231	0	0	78	4	0.75	0.98	0.85
Total	Our system	446	315	297	5	5	139	8	0.67	0.95	0.79
	BYU system	446	315	310	0	1	135	4	0.70	0.98	0.81

5 CONCLUSIONS AND FUTURE WORK

In this paper, we describe an approach to data extraction from the web that: is *unsupervised*; can *automatically segment data records* in web pages; and can *automatically label fields* in those records. This approach uses a novel method for automatically extracting domain-specific taxonomies from web pages. We evaluated both our method for acquiring taxonomies, and our data extraction method. We demonstrated that our approach achieves overall performance comparable to approaches that use hand-created ontologies, with much lower development costs. Our approach is also easily extensible to new domains.

In future work, we plan to improve our approaches to text segmentation and field labeling. For example, we may use a chunker to segment text, and use word sense disambiguation for field labeling. We may also use our taxonomies to improve text segmentation. For example, if we know that the term “Canon” appears only in the “Manufacturer” field, and “8MP” only in the “Megapixels” field, we can split “Canon X300 8MP Camera” into two text segments.

Our approach applies to feature tables with two columns only. In current work, we are looking at building ontologies and extracting data from multi-column tables.

6 ACKNOWLEDGEMENTS

We would like to thank our evaluators and the three anonymous reviewers. This work was supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.

References

1. Yi, L., Liu, B., Li, X.: Eliminating noisy information in web pages for data mining. In: Proceedings of KDD 2003. (2003)
2. Hsu, C., Dung, M.: Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems* **23**(8) (1998) 521–538
3. Kushmerick, N.: Wrapper induction: efficiency and expressiveness. *Artificial Intelligence* **118** (2000) 15–68
4. Pinto, D., et al.: Table extraction using conditional random fields. In: Proceedings of SIGIR'03. (2003)
5. Muslea, I., et al.: A hierarchical approach to wrapper induction. In: Proceedings of the Third International Conference on Autonomous Agents. (1999)
6. Chang, C., Lui, S.: IEPAD: information extraction based on pattern discovery. In: Proceedings of WWW 2001. (2001)
7. Crescenzi, V., Mecca, G., Merialdo, P.: RoadRunner: Towards automatic data extraction from large web sites. In: Proceedings of VLDB'01. (2001)
8. Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In: Proceedings of SIGMOD 2003. (2003)
9. Zhai, Y., Liu, B.: Web data extraction based on partial tree alignment. In: Proceedings of WWW 2005. (2005)
10. Mukherjee, S., et al.: Bootstrapping semantic annotation for content-rich HTML documents. In: International Conference on Data Engineering (ICDE), 2005. (2005)
11. Wang, J., Lochovsky, F.: Data extraction and label assignment for web databases. In: Proceedings of WWW 2003. (2003)
12. Etzioni, et al.: Web-scale information extraction in KnowItAll. In: Proceedings of WWW 2004. (2004)
13. Zhai, Y., Liu, B.: Extracting web data using instance-based learning. In: Proceedings of WISE 2005. (2005)
14. Embley, D.: Toward semantic understanding: an approach based on information extraction ontologies. In: Proceedings of the Fifteenth ADC. (2004)
15. Ding, Y., Embley, D., Liddle, S.: Automatic creation and simplified querying of semantic web content: An approach based on information-extraction ontologies. In: Proceedings of ASWC 2006. (2006)
16. Cai, D., Yu, S., Wen, J., Ma, W.: VIPS: a vision-based page segmentation algorithm. Technical report, Microsoft Research (2003)