Volume 2, No. 2, Mar-Apr 2011
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Extracting Information from Template Based Web Pages using Attribute Matching Patterns and Position Details

B.Venkat Ramana*
Computer Science Department, MIPGS, HYD
Research Scholar JNIAS, JNTUH
Hyderabad, India
venkatbhavanasi@gmail.com

Prof A. Damodaram
Department of Computer Science
JNTUH
Hyderabad, India
damodarama@jntuh.ac.in

Abstract: To extract structured data from web sites we recommend a new method for information extraction from web, which effectively uses content redundancy on the web. To start with, we extract records from the initial web sites and populate the seed database with the records. For a new extracted record, our method will compare it with the already available records in the seed database. We define a new matching technique that helps to match records with deferent representations across the sites. This new method finds the matching pattern between the attribute values of the two sites and ignores unwanted portions of the attribute. We developed an algorithm to find the attribute position details with sufficient matching values across pages. Finally we have done some experimental study with web data to know the effectiveness of our extraction approach.

Keywords: Information Extraction; Template WebPages; Pattern Matching; Data Position; Content Matching.

I. INTRODAUCTION

Web pages can be broadly classified as static and dynamic pages. The content and style of static pages does not change often. Where as the dynamic pages are generated when they are requested and change their content very often. Now a day's most pages on the web are generated by loading fixed page templates with data from a backend database. Previous studies estimate that as high as 45-55% of the content on the web is template content [1]. Youtube.com, amazon.com, cnn.com are some of the popular template based web sites.

Template based pages contain information about real world entities like products (name, price, description, etc.), books (title, author, price, publisher, etc.), restaurants (name, address, phone, etc.), business (address, phone, contact-person, etc.). Extracting and integrating such information from different web sites and populating seed database with this information, so that users can query for required information is an interesting and demanding problem in the field of web information extraction. Our idea is to extract records from all the template based sites that belong to a specific group such as restaurants, books, business which contains attribute values for a specific real world entity.

In this paper we present a new approach to extract records from multiple template based websites. Our method delivers high accuracy with minimal human intervention. One of the properties of template based sites is that multiple sites contain pages for the same entity [2]. Further, the values of attributes across the various pages for an entity are similar. It implies that there is redundant content across websites. We use this content redundancy by using extraction form one site to identify attribute values in the pages of overlapping entities in another site. The other property is that pages with in a website have a similar structure conforming to a common template. It implies that attribute values occur at fixed positions within pages of a site. Once we find the attribute value in a few pages, easily we can find the positions, and use this to extract values form other pages of the site. as well, for math, etc.

II. EASE OF USE

A. *Our Approach*

From a few initial sites we populate the seed database S of records. By using wrapper learning method human editors annotate attribute values in a sample page for each site. Since the sites are template based, a new site Xi which belongs to the group of similar sites, X will have same type of entities that are there in the seed database. Now we can scan the new site to find values that match attribute values of the records. The matching values are used to learn wrappers that are subsequently used to extract records from remaining other pages. This process is repeated for other new sites. This approach is largely un-supervised except for a few initial sites. By continuously expanding the seed database it ensures that sufficient overlap between the seed database and new sites.

Our extraction approach is based on the two properties that are mentioned above. The challenge to follow the first property is that, in practical, the entity attribute values can vary between sites due to data entry errors, abbreviations, deferent representations, etc and during extraction, this can lead to attribute vale match going undetected. The second challenge is that most of the web pages contain noisy and unwanted values that can lead to incorrect attribute value matching. These problems can be explained by an example, inspired by a real-world data.

The seed database R in figure-1(a) contains two records r1 and r2 with details name and address of the restaurants in city Hyderabad. Now the two restaurant pages p1 and p2 extracted from a new website represents the same information that is there in seed database R in almost similar way. The record r1 and page p1 refer to the same real world entity. The name attributes, "Hotel Holiday In" is spelled as "Hotel Holiday Inn" in pages p1 and p2. Similarly in the address attribute, the term "Street" is abbreviated to "St" in p2. Now if we use Jaccard coefficient [3] to measure the similarity between attribute values, where each value is treated as a block of words with space as a delimiter.

For any given two sets s1 and s2 the Jaccard similarity is defined as JC (s1, s2) = (|s1 ∩ s2|)/(|s1 ∪ s2|) Due to the different conventions for recording the address attribute, the Jaccard similarity between the address values in r1 and p1 belonging to the same entity is only 6 /13, which is less than 0.5. In contrast, the nearest transit value at the bottom of p1 has a higher similarity score of 4 /8 with the address value in r1. Similarly, the name values in r1 and p1 have a similarity score of only 1 /3, while the string "Hotel Holiday In" under "Related Hotels:" in p1 has a similarity of 1 with the name value in r2. Thus, extraneous values at the bottom of p1 have much higher similarity scores with name and address values in the seed records which can lead to false positive matches.

Record	Name	Address
r1	Hotel Holiday In	Road # 3, RP road, Hyderabad, India-500001
r2	Jasmines Restaurant	Road # 12, Street # 6, Malakpet, Hyderabad, India-500035

(a) Seed Database R

Hotel Holiday Inn
Road#3, RP Road,
(between 1st and 4th cress)
Hyderabad, India-500001

Related Hotels:
Jasmines Restaurant
Swagat Group of Hotel

(b) Page p1

Jasmines Restaurant
road #12, St #6,
(between 4th and 8th cross)
Malakpet, Hyderabad,
India-35

Related Hotels:
Hotel Holiday Inn
Swagat Group of Hotels

(c) Page p2

Figure: 1 (a) Seed Database; (b) Page p1; (c) Page p2.

B. Our Contibutions

The above mentioned problems can be overcome by a new extraction approach that uses content redundancy over the websites and structural similarity among template based pages to extract attribute values with high precision and with minimum human intervention.

To cope with different attribute formatting conventions across sites, we define a new similarity function. Our new function leverages the fact that attribute values in template-based pages have a templatized structure and uses this to improve matching accuracy. Our proposed metric discovers repeating patterns among the matching portions of attribute value pairs from two sites, and uses this to filter out non-matching portions when computing the similarity score between attribute value pairs. In Figure 1, our similarity function detects that the lines beginning with "(between" are extraneous across the address values in p1 and p2, and so ignores them when matching the values with addresses in the seed database. This boosts the similarity scores (Jaccard similarity = 3/ 4) between address values in pages p1 and p2, and their corresponding values in records r1 and r2, respectively, and ensures that they are matched correctly. Thus, our new similarity metric is able to match attribute values (with diverse representations) belonging to the same entity while keeping the number of false positive matches low. In order to further filter out noisy matches, we match values for multiple attributes and also exploit the fact that attribute values occur at fixed positions within the pages of a template-based web site. Thus, suppose that for a configuration of attribute positions, we define the support as the number of pages for which the values at the positions match seed record values. Then, we can essentially prune configurations with insufficient support. We propose an efficient [3] algorithm to systematically enumerate attribute position configurations with sufficient support. In Figure 1, our algorithm will prune the spurious match between the string "Jasmines Restaurant" under "Related Hotels:" in p1 and the name attribute in r2 since the address value in page p1 does not match the address value in record r2 for restaurant "Jasmines Restaurant".

We conduct an extensive experimental study with real-life web datasets from different verticals. Our extraction approach performs well delivering greater than 96% precision and more than 82% recall for extracted records from a wide range of web sites. For our similarity model we treat each attribute value string as sequence of words separated by special characters for example, space, tab, comma, hyphen are considered as word delimiters. The similarity metric is designed such a way that is robust to typographical errors, abbreviations, word re-orderings, and word normalization, etc. There are other proposals for matching similarity functions have been proposed in the research literature.

The Jaccard coefficient and Cosine similarity metrics [3, 9], Extractions to use q-grams instead of words [7], and Edit distance family of functions [6, 8, 9, 5]. Sim (x, y) is used to denote similarity between strings x and y which are values of attribute a. The values x and y are weakly similar if sim (x, y) ≥ Tw where Tw is the weak similarity threshold. A variant of the "Cosine similarity over q-grams" similarity function [7] can handle both spelling errors as well as word rearrangements.

Our algorithm takes the advantage of similar content between the seed database R and the web pages of sites in W for structured data extraction. For the DOM tree representation of a web page p and a node n in the page p, let x be the unique path from the root to n in the DOM tree. The path x is the position of node n in page p. Further, p[x] denotes the value of the node n at position x in page p. For a leaf node, the value is essentially the text string contained in it. If n is an internal node, then its value is the concatenated sequence of text from the leaves of the subtree rooted at n (in the order in which the nodes appear in the DOM tree). For a seed record r in R, we denote the value of attribute ai ∈ A in r by r[ai]. With each record r, we associate the web site from which the record was extracted, denoted by W(r). The extraction algorithm scans the pages of a new web site W to find node values that match attribute values in the seed database R, and uses this to infer the node position for each attribute within the pages of W. It then extracts entity records from pages of the web site by extracting the attribute values at the identified positions. A key challenge here is to find the correct node values within a page that match the attribute values within a seed record. Even with strong similarity, due to noise and extraneous information in web pages, we may still have spurious matches within a page. For instance, in Figure 1, values like "Jasmines Restaurant" under "Related Hotels:" can match name values in seed records. In order to compute attribute positions, given a seed database R, web

site W, and a minimum support parameter β, find a maximal set S of (attribute, position) pairs such that sup(S) ≥ β.

Notice that we are looking to find positions for as many attributes as possible and not necessarily all attributes – this is to accommodate scenarios in which certain attributes are missing from site W. In case there are multiple sets S with the same number of (attribute, position) pairs and with support at least β, then we select the set S with the maximum support. In our experiments (see Section 4), we found that setting the support parameter β = 10 is effective at filtering out the spurious matches. Procedure FINDATTRPOS describes an efficient algorithm for computing the maximal set S of (attribute, position) pairs with support ≥ β for a new web site W. Similarity computation between a node value and an attribute value in a record serves as a basic building block for determining record-level similarity between attribute values in a page of W and a seed record from R. We keep track of the weakly similar (page, position) and (record, attribute) pairs by storing in WS(a, x) the record, page pairs (r, p) such that r[a] is weakly similar to p[x].

For each (attribute, position) pair (a, x) such that there are a sufficient number of pages with weakly similar values in position x, we compute strong similarity scores between (r[a], p[x]) pairs. We store in SS(a, x) the (record, page) pairs (r, p) such that r[a] is strongly similar to p[x]. Note that SS(a, x) will contain fewer false positive matches compared to WS(a, x). Due to spurious attribute value matches still contained in SS(a, x), within each page of W, there may be multiple attribute position configurations for which values in the page match (portions of) a record in the seed database R. Across all the pages of W, the number of these attribute position configurations could become really large, even though most of them do not have the required support.

Computing support for each of these configurations to determine the maximal configuration with support ≥ β can turn out to be very inefficient. Instead, we devise an efficient algorithm based on the following observation: for a pair of (attribute, position) pairs sets S, S′, if $S \subseteq S'$, then $sup(S') \leq sup(S)$. Thus, we can use an iterative algorithm that generates candidate (attribute, position) pair sets of size k in the kth iteration and stores these in Ck. Candidate sets whose support is less than β are pruned from Ck since any superset of these cannot have support ≥ β. The remaining sets in Ck (after pruning) are used to generate supersets of size k + 1 which become candidates for the next iteration. Once we have identified the maximal set S of (attribute, position) pairs with support ≥ β, we extract entity records by extracting attribute values at the specified positions in S from the pages of web site W.

III. EXPERIMENTAL EVALUATION

In this section, we present experimental results with real-life web datasets which demonstrate the effectiveness of our content matching-based extraction approach. Specifically, we show that our strong similarity metric and multi-attribute matching technique result in high-precision extractions while ensuring adequate web site coverage. Our experimental setup includes: **Datasets:** We use two real-life datasets covering two verticals: restaurant and bibliography. Each dataset consists of a set of seed records and crawled pages from a set of test sites. We use the seed records to extract from the single-entity pages belonging to each of the test sites, and report the precision and coverage of the extractions.

We classify attributes into core and non-core. Core attributes are present in every page belonging to the test dataset, while noncore attributes are optional. The seed data for restaurants is obtained from chefmoz.com. Data from chefmoz.com is available as structured data in RD F format. Extractions are performed on 17 sites. The attributes extracted are: 1. restaurant name (core) 2. address (core), 3. phone, 4. payment, and 5. cuisine. The seed dataset consists of 40000 records randomly selected from chefmoz data. For the bibliography dataset, we use data from DBLP as seed records. DBLP data is available in XML format. The seed dataset consists of 40000 records randomly selected from this dump. 7 sites are used as test sites. The following attributes are selected from the DBLP dataset: 1. title (core), 2. Author (core) and 3. Source.

Metrics: We use precision and coverage as the primary metrics to evaluate the quality of the extractions. Since the datasets we use are very large, generating the complete ground truth editorially is a daunting task. Hence, we choose a random set of 1000 pages from each dataset, and generate the ground truth for this set. Precision metrics are reported on this random set. We define the coverage for a dataset as the fraction of pages in the dataset from which we are able to extract core attributes.

Extraction Schemes: We use the FINDATTRPOS procedure described in Section 3 to compute the attribute positions (from which values are extracted) for each test site. We set the support parameter α for strong similarity computation to 0.1, and use β = 10 to prune attribute position configurations with inadequate support. We fix the weak similarity threshold Tw at 0.5, and vary Ts, the strong similarity threshold, between 0.5 and 0.9 in our experiments.

In order to compare the quality of extractions using weak and strong similarity, we consider a variant of FINDATTRPOS which we refer to as FINDATTRPOSW. FINDATTRPOSW is identical to FINDATTRPOS except that it uses weak similarity (instead of strong similarity) to determine the matching attribute values between seed records and web pages. Thus, in FINDATTRPOSW, for a set S of (attribute, position) pairs, sup(S) is the number of distinct pages in $\cap_{(a,x)\in SWS(a,x)}$. We set the minimum support parameter β to 10.

Platform: All the experiments were performed on a shared Hadoop 0.20 cluster. The execution times reported are based on the number of map/reduce tasks and the average time of the map/reduce tasks.

A. *Experimental Results*

In order to gauge the impact of strong similarity, we compare the precision and coverage of extractions generated by FINDATTRPOS and FINDATTRPOSW.

Table 1: Dataset summary.

Dataset	*#Seed Records*	*# attributes*	*#Test Sites*	*# Pages*
Restaurant	**40000**	**5**	**17**	**984992**
Bibliography	**40000**	**3**	**7**	**1299329**

Table 2: Precision of Extractions for all attributes.

Restaurant		*Bibliography*	
Attribute	*Precision*	*Attribute*	*Precision*
Name	*78.26*	*Title*	*96.14*
Address	*99.74*	*Author*	*98.12*
Phone	*100.00*	*Source*	*100.00*

Payment	***100.00***	-	-
Cuisine	***100.00***	-	-

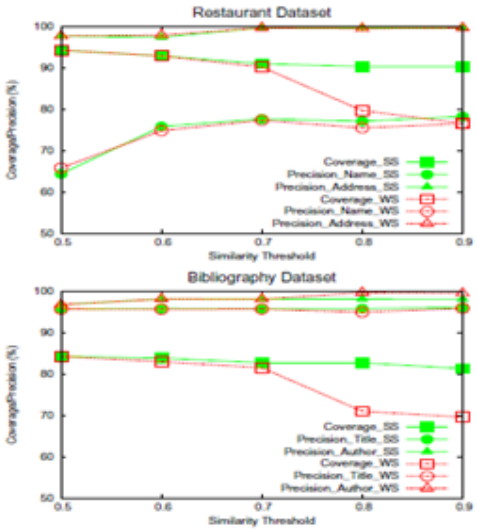

Figure 2: Precision/coverage of weak and strong similarity based extraction of core attributes.

Figure 2 plots these for the 2 datasets as Ts for FINDATTRPOS and Tw for FINDATTRPOSW are increased from 0.5 to 0.9 (Tw for FINDATTRPOS is fixed at 0.5). In the plots, we use suffixes SS and WS to qualify the precision and coverage metrics of procedures FINDATTRPOS and FINDATTRPOSW, respectively. It can be seen that the extraction precision increases with the similarity threshold for both the techniques. There is a significant coverage drop for FINDATTRPOSW at high (> 0.7) weak similarity threshold values.

Strong similarity-based matching, on the other hand, provides both high precision and high coverage at higher strong similarity threshold values (> 0.7). This is because strong similarity boosts the similarity scores between diverse representations of the value of an attribute for the same entity which otherwise have low weak similarity scores. As a result, at the higher threshold values, true matches are retained (leading to high coverage) and false matches are pruned (leading to high precision). In fact, it is interesting to observe that FINDATTRPOS consistently has high coverage over the entire range of Ts values between 0.5 and 0.9.

Figure 3 plots the strong and weak similarity scores for 200 address pairs between the seed database and the test sites. All of the address pairs have weak similarity scores exceeding 0.5, and are classified by hand into true and false matches. It is easy to see that the weak similarity scores of both true as well as false matches are distributed between 0.5 and 0.9. Thus, with weak similarity, there are true matches with low scores and false matches with high scores. This makes it difficult to find a threshold value that cleanly separates the true matches from the false ones. In contrast, for several true matches, the strong similarity scores are boosted close to 1 even from very low weak similarity scores. Thus, with a high enough strong similarity threshold ($\approx$ 0.9), we can identify the true matches while filtering out the false ones.

Figure 3: Scatter plot of strong similarity vs weak similarity scores. True matches are shown as green triangles and noisy matches as red squares.

As we look at the precision metrics for the entire core and optional attributes for both the datasets (see Table 2). We only consider strong similarity for extraction, and set the parameter values Ts = 0.9 and Tw = 0.5 in our FINDATTRPOS extraction procedure since these settings yield the best results across the 2 datasets. As can be seen, the precision for most of the attributes is above 95% and the coverage of core attributes for both the datasets exceeds 80%.

The precision of the name attribute in the restaurant dataset is somewhat low because of the presence of long lists of "Nearby Restaurants" in the restaurant pages of the website; these results in false matches with the name attribute values in seed records.

In order to quantify the amount of filtering achieved due multi-attribute matching in procedure FINDATTRPOS, we track the number of attribute positions in the generated candidate sets Ck. Let Ck(a) be the set of distinct positions (DOM tree paths) for an attribute a in Ck. The decay in the number of distinct positions |Ck(a)| as a function of k indicates the efficiency of multi-attribute matching. Table 3 lists the number of distinct paths for the core attributes in both the datasets as a function of k. Observe that a majority of the attribute positions involved in spurious matches are pruned within two iterations. This indicates that considering pairs of attributes when matching values can substantially improve matching accuracy.

Table 3: Number of Positions for core attributes in multi-attribute matching.

	Restaurant		***Bibliography***	
	Name	***Address***	***Author***	Title
K=1	**1002**	**387**	**254**	**694**
K-2	**113**	**64**	**34**	**39**

Table 4 shows coverage of strong similarity-based extractions at Ts = 0.9 for the restaurant dataset as the number of seed records is increased from 2000 to 40000. As can be seen, coverage jumps from 53.57% to 90.37% due to higher content overlap between seed records and web pages at the larger seed set sizes.

Table 4: Coverage of Extraction Vs Seed set size for Restaurant data.

Seed Size	**2000**	**5000**	**10000**	**20000**	**40000**
Coverage(%)	**53.57**	**59.73**	**61.06**	**69.40**	**90.37**

Table 5: Running Time(Hrs) of the diffirent steps for the two data sets.

Stage	*Restaruant*	*Bibilography*
Weak Similarity	**23987**	**17488**
Strong Similarity	**163**	**15**
Multi-attribute Matching	**22**	**5**
Extraction	**65**	**26**
Total	**24237**	**17534**

Table 5 provides the execution times of different stages. It can be seen that weak similarity computation dominates the execution time despite the use of prefix filtering [7] in our implementation. A complete run involving all the stages can be done on a 1000 CPU cluster in 1 day for the restaurant dataset and in 0.73 days for the bibliography dataset. Note that the execution time for the restaurant dataset is more than that for the bibliography dataset. The reason is that pages in the restaurant dataset are structurally more complex than those in the bibliography dataset: the average number of nodes per page for the restaurant dataset is 2.5 times more than that for the bibliography dataset.

IV. RELATED WORK

In recent years, a number of research papers [16, 20, 9, 25, 14, 11, and 13] have studied the problem of extracting structured data from web pages. Early proposals for extracting structured data from the web were based on wrapper induction [16, 20]. These require human editors to annotate pages from each site and thus have high overhead. In recent years, there has been a flurry of research activity on extraction techniques that incur little manual effort. [9, 25, 11] devise methods to detect repeated patterns of tags within a web page and use this to extract records from the page.

In [5, 1], attribute models based on Hidden Markov Models (HMMs) are learnt from training data, and these are used to segment short text strings like addresses and bibliographic entries. Web pages have a more complex hierarchical structure, and Zhu et al. [14] propose Hierarchical Conditional Random Fields (HCRFs) to label attribute values in web pages. An HCRF is a graphical model that captures both hierarchical and sibling dependencies in the tree corresponding to a web page.

Markov Logic Networks (MLNs) [21] go a step further and allow relationships between arbitrary tree nodes to be expressed as first-order formulas. MLNs are used in [13] to extract structured data from web forum sites. These models rely on structural features of pages (e.g., phone numbers follow address values) and content features of attributes (e.g., 5-digit numbers correspond to zip codes) to label attribute values.

The precision of machine learning models may be poor in web environments due to the heterogeneity in web page structure and attribute content formats, and noise in web pages. Our extraction approach overcomes these problems by exploiting content redundancy across sites, and uses the actual extracted attribute values to find matching values within web pages. Thus, we circumvent the difficult problem of building models that can capture the diverse structural and content formats prevalent across web sites.

Finally, there is a body of work on iteratively growing seed sets of relation instances and patterns for relation extraction. This is done by finding occurrences of the seed data in the corpus, discovering patterns, and matching the patterns to augment the seed data. The use templatized page and content structure in a site distinguishes our approach from these techniques.

Basically our extraction approach depends on being able to approximately match the attribute values for an entity across multiple sites. Fortunately, a number of approximate string matching algorithms have been proposed for detecting duplicate records in databases, text searching in the presence of spelling errors, etc. Comprehensive surveys of approximate string matching techniques can be found in [10, 15]. Existing similarity functions for string matching take as input two strings, and return a similarity score that quantifies the match between them.

A popular measure used to gauge the similarity between two strings is the string edits distance. The edit distance metric works well for typographical errors but it cannot capture word rearrangements, insertions, and deletions. To address this, numerous variants of the edit distance metric have been proposed in the literature like affine gap distance [24] that allows gap mismatches, block edit distance [18] that allows word moves, and a fuzzy match similarity function that allows words to be inserted/deleted with a cost equal to the IDF weight of the word [22].

However, most variants either do not handle word rearrangements well, or are too expensive from a computation perspective. For instance, finding the exact block edit distance between two strings is an NP-hard problem [18]. TheWHIRL [8] system adopts a different approach based on Cosine similarity between IDF-weighted words which it borrows from the IR literature. Unfortunately, while Cosine similarity can handle word swaps and weighs words based on their importance, it is less resilient to word misspellings. To alleviate this problem, Gravano et al. [17] propose a similarity metric that computes the Cosine similarity between IDF-weighted q-grams (instead of words). This metric has a number of desirable properties – it is capable of handling both word re-orderings as well as spelling errors, and is computationally efficient.

Our notion of weak similarity also employs q-grams and is a variant of the similarity function proposed in [17]. Further, our strong similarity metric adds a new dimension by also taking into account the template structure when matching strings. Unlike previous similarity functions, it takes as input two sets of string values, and refines similarity scores based on the matching pattern between value pairs from the two sets. A bulk of the previous work has focused on using the above string similarity functions to match records with multiple attributes. [4, 23] train classifiers to combine the multiple attribute-level similarity scores into a single record-level similarity score, while [17, 22] simply extend the edit distance and Cosine similarity variants to work at the granularity of records as opposed to individual attributes. In contrast, in our web extraction scenario, we are interested in finding values within unstructured web pages that match attribute values within a record. Our problem setting is a lot more challenging due to the presence of noise in web pages; our proposed solutions filter out the noisy attribute value matches by exploiting the template structure of attribute content and web pages.

V. CONCLUSION AND FUTUREWORK

To extract structured data from web sites we recommend a new method for information extraction from web, which effectively uses content redundancy on the web. To start with, we extract records from the initial web sites and

populate the seed database with the records. For a new extracted record, our method will compare it with the already available records in the seed database. We define a new matching technique that helps to match records with deferent representations across the sites. This new method finds the matching pattern between the attribute values of the two sites and ignores unwanted portions of the attribute.

In this paper, we proposed a new approach that exploits overlapping content across web sites and the template structure of web pages to extract structured data from the web. We defined a new similarity metric for matching previously extracted attribute values with the content in a fresh page. Our new metric takes into account the matching pattern between attribute values from two sites to refine similarity scores for differently formatted attribute values belonging to the same entity. We also developed an Apriori-style algorithm for efficiently enumerating attribute positions with matching values in a sufficient number of pages. In our experiments with real-life web data sets, our techniques were able to extract records with > 95% precision and > 80% recall. An important direction for future work involves extending our methods to handle non-text numeric (e.g., price) and image (e.g., ratings) attributes.

VI. REFERENCES

[1] E. Agichtein and V. Ganti. "Mining reference tables for automatic text segmentation". In Procedingd of SIGKDD, v.20 n.1, p.152-187 ; 2004.

[2] E. Agichtein and L. Gravano. "Snowball: extracting relations from large plain-text collections." In ACM DL, In Proceedings of the 5th ACM International Conference on Digital Libraries (ACM DL), pages 85–94 ; 2000.

[3] R. Agrawal and R. Srikant. "Fast algorithms for mining association rules." In Procedings of SIGMOD-VLDB 1994, pages 487–499;1994.

[4] M. Bilenko and R. Mooney. "Adaptive duplicate detection using learnable string similarity measures." In Procedings of SIGKDD, pages 39-48 ;2003.

[5] V. Borkar, K. Deshmukh, and S. Sarawagi. "Automatic segmentationof text into structured records." In SIGMOD, In Proceedings of the. ACM-SIGMOD International Conference on Management of Data, pages 175–186;2001.

[6] S. Brin. "Extracting Patterns and Relations from the World Wide Web".In WebDB, 172-183. 7, Electronic Edition ; 1998.

[7] S. Chaudhuri, V. Ganti, and R. Kaushik. "A primitive operator for similarity joins in data cleaning". In ICDE, In Proceedings of. ICDE ;2006.

[8] W. Cohen. "Integration of heterogeneous databases without common domains using queries based on textual similarity". In SIGMOD, ACM SIGMOD Record, v.27 n.2, p.558-560; June 1998.

[9] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: "Towards automatic data extraction from large web sites". In VLDB, v.14 n.4, p.197-214, December ;2001.

[10] A. Elmagarmid, P. Ipeirotis, and V. Verykios. "Duplicate record detection: A survey". IEEE TKDE, no. 1, pp. 1-16, Jan. 2007, doi:10.1109/TKDE;2007.

[11] G. Miao et al. "Extracting data records from the web using tag path clusterting". In Procedings of WWW, 2009.

[12] D. Gibson, K. Punera, and A. Tomkins. "The volume and evolution of web page templates". In WWW, Procedings of the 14th International Conference on World Wide Web, pages 830-839, New York, NY, USA; 2005.

[13] J. Yang et al. "incorporating site-level knowledge to extract structured data from web forums". In Procedings of WWW, 2009.

[14] J. Zhu et al. "Simultaneous record detection and attribute labeling in web data extraction". In SIGKDD, Proceedings of the 15th ACM SIGKDD international conference on Knowledge pages:494-503;2006.

[15] N. Koudas, S. Sarawagi, and D. Srivastava. "Record linkage: Similarity measures and algorithms". In SIGMOD (Tutorial), ISSN : 0975-3397 Vol. 3 No. 3 Mar 2011,pages 802-803; 2006.

[16] N. Kushmerick, D. S. Weld, and R. Doorenbos. "Wrapper induction for information extraction". In IJCAI, IJCAI-97, Pg:729-735;1997.

[17] L. Gravano et al. "Text joins in an RDBMS for web data integration".In WWW, pages 729–731; 2003.

[18] D. Lopresti and A. Tomkins. "Block edit models for approximate string matching". Theoretical Computer Science, 181(1), Theoretical Computer Science, v.181 n.1, p.159-179, July 15, 1997 ..... Proceeding of the 18th ACM conference on Information and knowledge;1997.

[19] C. Manning, P. Raghavan, and H. Schutze. "Introduction to Information Retrieval". Cambridge University Press, First IEEE International Conference on Semantic Computing , pages 19–26;2008.

[20] I. Muslea, S. Minton, and C. Knoblock. "Hierarchical wrapper induction for semistructured information sources". Autonomous Agents and Multi-Agent Systems, 1(2), International Journal of Cooperative Information Systems 10(1-2):145-169 ;2001.

[21] M. Richardson and P. Domingos. "Markov logic networks". Journal of Machine Learning,62(1), 62:107–136 ;2006.

[22] S. Chaudhuri et al. "Robust and efficient fuzzy match for online data cleaning". In SIGMOD, Proceedings of the ACM SIGMOD;2003.

[23] S.Sarawagi and A. Bhamidipaty. "Interactive deduplication using active learning". In SIGKDD, EDBT 2009: 450-461 .... SIGKDD Explorations 6(2): 61-66 (2004);2002.

[24] M. Waterman, T. Smith, and W. Beyer. "Some biological sequence metrics". Advances in Math., 20(4), vol. 20, no. 4, pp. 367-387;1976.

[25] Y. Zhai and B. Liu. "Web data extraction based on partial tree assignment". In procedings of WWW conference, vol. 20(4), no. 4, pp. 467-478;2005.

[26] Pankaj Gulhane Rajeev Rastogi Srinivasan H Sengamedu Ashwin Tengli- Yahoo! Labs, Bangalore Microsoft IDC, Bangalore.- "Exploiting Content Redundancy for Web Information Extraction". The 36th International Conference on Very Large Data Bases, September 13-17,2010, Singapore; Proceedings of the VLDB Endowment,Vol.3,No. 1, Pg:578-587;2010-VLDB,21508097/10/09- 2010.