Journal of Theoretical and Applied Information Technology
31st March 2012. Vol. 37 No.2

ISSN: 1992-8645www.jatit.org E-ISSN: 1817-3195

DEVELOPMENT OF AN EFFICIENT VERTEX BASED TEMPLATE EXTRACTION TECHNIQUE FOR WEB PAGES

[1]N.P.K. GANESH KUMAR, [2]Dr.N.K. SAKTHIVEL

[1]PG Student, School of Computing, SASTRA University, Tamil Nadu, India.

[2]Professor,School of Computing, SASTRA University,Tamil Nadu, India.

Email: [1]npkganesh@gmail.com,[2]sakthi@cse.sastra.edu

ABSTRACT

World Wide Web is useful source of information. Websites contains many web pages which automatically occupied by common template with contents. In former system, TEXT template extraction technique is used for template detection and also extract template from dynamic web pages, but it extracts the entire site and stores in database. It provides the unnecessary information for further processing. To overcome this problem, Vertex based information extraction technique is used, which would improve the web search results quality and web integration for dynamic web pages. To operate at web scale, Vertex employs a host of novel algorithms for web page clustering, robust wrapper learning, detecting site changes, and rule relearning optimization. The system is deployed in production and currently extracts millions of records from many websites, in order to get high efficiency. Vertex is the first system to do high accuracy information mining at web scale.

Keywords: *Clustering, Xpath Rule, Template Extraction.*

1. INTRODUCTION

World Wide Web (WWW) is the most useful source of information.It is easy to access the information present in the World Wide Web. The unknown templates are considered to be harmful for the machines.The reason is they degrade the accuracy and performance due to irrelevant terms present in the template. To reduce this TEXT template extraction technique [2] is used. Templates can be detected and extracted automatically from heterogeneous WebPages.It extracts the entire site and stores in database for static WebPages.To prevent this we use vertex based template extraction technique. The vertex system extracts the index terms and works on the dynamic web pages and it would improve the web search quality and web integration etc. The system is deployed in production and currently extracts millions of records from many websites.Vertex is the first system to do high accuracy information extraction at web pages.

Vertex [1] is a system developed at Yahoo for extracting structured records from template based WebPages.As an example, consider the page shown in Fig.1 for restaurant " Inn Kensington " from the aggregator web site www.yelp.com, the page contains wealth of information including details like restaurant name, water service, and delivery, has TV, parking, alcohol etc.Vertex extracts this information from such detail pages and stores the extracted data for each page as attribute of records.This is shown in table1.

Table1. Attribute of Records

Name	**Water service**	**Parking**	**Good for kids**	.....
Inn Kensington	Yes	Street	Yes	.....
Paulie's pickling	No	Street	Yes	.....
Pie tisserie	No	No	Yes	
Pie Fridays	No	Street	Yes	

Journal of Theoretical and Applied Information Technology
31st March 2012. Vol. 37 No.2

ISSN: 1992-8645www.jatit.org E-ISSN: 1817-3195

Fig 1. Example Restaurant detail page

Vertex brings together various technology components incorporating novel algorithms to handle the complete extraction lifecycle, from clustering pages within a websites, to learning extraction rules, to detecting site structure changes, and finally relearning broken rules.In this paper, we describe the architecture and implementation details of the vertex system.To operate at web scale, vertex relies on a host of algorithmic innovations,

1) Clustering algorithm for grouping similar structured pages in a web site. Our algorithm makes only 3 passes over the data.

2) Greedy algorithm for picking structurally diverse sample pages for annotation.

3) Apriori style algorithm for learning very general XPath -based extraction rules that are robust to variations in site structure.

4) Site changes detection scheme that monitors a few sample pages per site, and subjects the pages to different structural and content tests. 5) Algorithm for optimizing editorial costs by reusing rules.

2. RELATED WORK

The process starts from template detection and then extraction will be done.The template extraction problem can be categorized into two broad areas. The first area is the site level template detection where the template is decided based on several pages from the same site. Previously only tags were considered to find templates [6] but any word can be a part of the template or contents.It considering document as trees but the operations on tree is usually too costly to be applied to a large number of documents.The other area is the page level template detection where template is computed within a single document. It represents web document as matrix and find cluster within the matrix. Bi clustering or co clustering is another clustering technique to deal with a matrix. [7], [8].Co clustering algorithm find synchronized clustering of the rows and columns of a matrix and need the number of columns and rows as input parameter.However, we cluster only documents not paths, and moreover, the number of clusters of columns and rows are unknown.

Early work on wrapper induction falls into two broad categories: global page description or local landmark-based approaches (e.g. [4], [5]) detect repeated patterns of tags within a page in an unsupervised manner, and use this to extract records from the page.

The template extraction from the heterogeneous web pages [2] has two disadvantages. The first failure is, it considers only static web pages, for example the news web site is uploading daily but in this case we can access only the present day's news. The second failure is it extracts the entire site, for example advertisements, navigation panels, headers, footers, and copyright information etc.., is also extracted. To prevent this we use vertex based template extraction technique. The vertex based technique extracts the index terms and works on the dynamic web pages.

3. VERTEX SYSTEM

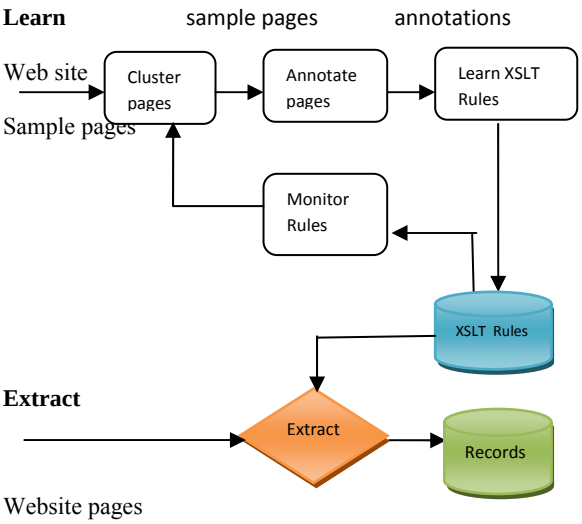

Fig 3. Vertex system architecture

3.1. Learning Subsystem

The learning subsystem is dependable for learning a new set of extraction rules for a specific

ISSN: 1992-8645www.jatit.org E-ISSN: 1817-3195

site. Rule learning occurs in two contexts :(1) initially, when rules are learnt for site for the first time, and (2) consequently, when rules are relearns for the site once a change in its structure is detected. In the afterwards relearning scenario, certain optimizations like leveraging existing rules to reduce editorial effort are applicable. The key components in the learning system are as follows:

3.1.1 Page clustering

A single website may contain pages compliant to multiple different templates. We identify these different groups of template based pages by clustering the pages within the sites.A sample of pages from each web site is first together.A shingle-based signature [3] is computed for each web page based on HTML tags (and not content) in the page, and the pages are clustered using the signatures.A single XSLT rule is learnt for each cluster containing pages with similar structure. The clustering component starts by collecting sample pages P from the Web site for which rules is to be learnt.Our purpose is to group structurally similar sample pages mutually. With each sample page in P, we associate an 8-byte shingle vector signature which is computed as follows.We refer to a nearby cycle of l tags within the page as a shingle.

Let S be the set of all shingles in the page - this can be efficiently computed by sliding a window of length l over the tag sequence of the page. Observe that S captures the structure of the page. Further, if S and S' are the shingle sets for two pages p and p', then we can use the overlap between the shingle sets | S □ S'| / | S □ S'| as a measure of their structural similarity. Let h1, h8 be independent hash functions that map a shingle to a single byte. The value of the i^{th} byte in vector v is computed by applying hi to each shingle in S and selecting the minimum 8-bit hash value among all the shingles for the pages. More formally, v[i] = min $_{sh\varepsilon S}$ {hi (sh)}.For two pages p and p' with shingle vectors v and v' and shingle set S and S'. it can be shown that v[i] = v'[i] with probability | S □ S'| / | S □ S'|. We use masked shingle vectors to group all pages with shingle vectors than match on k out of the 8 byte values. A k/8 masked shingle vector v contains hash values for k indices and the remaining 8 - k indices are wild cards " * " that match any value. A masked shingle vector v covers another vector v' if for all indices i either v[i] = v'[i] or v[i] = *.

Vertex's novel clustering algorithm is described in algorithm 1.It makes three passes over the pages in P. In the first pass, for each page p, counts of all candidates 6/8, 7/8, 8/8 masked shingle vectors that covering the page's shingle vector v are incremented by 1.

Algorithm 1. Cluster pages

Input: sample pages P from web site to be clustered;
Output: Set of clusters;
/* First Pass */
Initialize hash table H to empty;
for each page p ε P **do**
Let v be the shingle vector for p,
for each 6/8, 7/8, and 8/8 masked shingle vector v'
covering v do
if v' is in hash table H **then**
Increment the count for v';
else
Insert v' with count 1 into H;
end if
end for
end for
/* second pass */
for each 8/8 vector v in H in increasing order of counts
do
Let v' be the masked shingle vector in H with maximum count covering v,
Decrement counts of all masked shingle vectors ≠
v' in
H covering v (by v's count);
end for
Delete masked shingle vectors with countless than
threshold from H;
/* Third pass */
for each masked shingle vector v ε H, C_v= □,
for each page p ε P with shingle vector in H with maximum count covering v,
Add p to C_v;
end for
return {(C_v, v, count for v) : v ε H};

There are $\sum_{i=0}^{2}.\binom{8}{i}$ candidate masked shingles vectors that cover each shingle vector v and these are obtained by masking 0, 1 or 2 values in v.Thus, at the end of the first pass, each candidate masked shingle vector in hash table H has a count equal to the number of page shingle vector it covers.In the second pass, for each 8/8 shingle vector in increasing order of counts, a

ISSN: 1992-8645www.jatit.org E-ISSN: 1817-3195

single candidate from the $\sum_{i=0}^{2}.\binom{8}{i}$ possible candidate that cover it is selected. The selected candidate is the one with the largest count in H, and the counts for the remaining candidates are decremented by the size of the 8/8 cluster. Thus the end of the second pass, each page shingle vector contributes to the count of only one masked vector in H. We delete all masked vector with negligible counts from H after all 8/8 clusters are assigned.Since candidate vectors counts continuously vary during the second pass, we perform a third pass to do the final assignments of pages to cluster without adjusting vector counts. In the third pass, each page p with shingle vector v is assigned to the cluster $C_{v'}$ forthe masked shingle vector v' in H that covers v and has the maximum count.

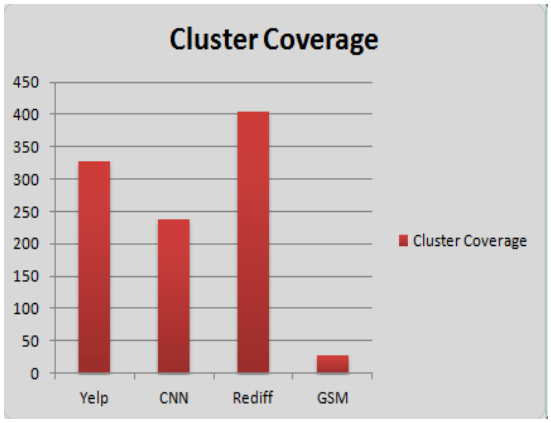

Fig 4. Cluster coverage

3.1.2. Page Annotation

While pages within the cluster have similar structure for the most part, they may contain minor structural variations due to missing attribute values, HTML tags, etc. From each cluster, a few sample pages that are structurally varied are selected for annotation by human editors.For an XPath X_i, let F (X_i) denote the frequency of Xi; that is, the number of cluster pages that contain Xi.In order to differentiate between informative and noisy XPaths, we assign different weights to them. For this, we leverage the fact that, in a particular web site, noisy sections share common structure and content, while informative sections differ in their actual content. The informative of an XPath X_i is determined as:

$$\mathbf{I(X_i) = I - \frac{(\sum t \varepsilon Ti\ F(Xi,t))}{M\ .|Ti|}}$$

Where T_i denotes the set of content associated with XPath Xi, F (X_i , t) denotes the number of pages containing content t at the node matching X_i , and M is the number of cluster pages.Spontaneouslyan XPath Xi in a noisy portion of the page will have repeating content across pages, thus will end up with a low informative score close to 0 since $|Ti| \approx 1$ and $\sum t \in TiF(Xi, ti) \approx M$. On the other hand, we will assign a higher informative score to an XPath belonging to an informative region that has distinct content across pages, here the informativeness score will be close to 1 since $\sum t \in TiF(Xi, ti) \leq M$ but $|Ti| \approx M$. Since we are interested in covering frequently occurring XPaths belonging to informative regions, we assign each XPath Xi a weight w (Xi) = F (Xi). I(Xi). Ideally, we would like our annotation sample to contain pages with the high weight XPaths since these have the attributes that we wish to extract. Thus, for a K size sample, our problem is to select K pages such that the sum of the weights of the distinct XPaths contained in the page is maximized. This is identical to the maximum coverage problem which is known to be NP-hard, but can be approximated to within a factor of 1- 1/e using a simple greedy algorithm. We can achieve this approximation bound by modifying the approximate greedy solution to order pages based on uncovered XPaths as shown below.

Algorithm 2: Weighted greedy algorithm

Input: cluster C = { p_1,.....,p_n}, and sample size K;

Output: K or less sample pages;

Initialize the uncovered XPath set X to all distinct
XPaths in C, and sample S to □;
While X ≠ □ and |s| ≤ K **do**
Find P_i = max $_{pj\varepsilon}$(C-S) {$\Sigma Xi \square pj \cap \mathcal{X}(Xi)$};
S = S □ {p_i};
x = x - {XPaths in p_i};
end while
return S;

3.1.3. XSLT Rule learning

The annotations in the annotated sample pages AS of a cluster specify the location of nodes containing attribute values. These are used to learn an XSLT rule for extracting attribute values from new pages that map to the cluster. The XSLT rule

ISSN: 1992-8645www.jatit.org E-ISSN: 1817-3195

contains XPath expressions that identify the node in the new page corresponding to each attribute and other components that extract the actual text value of interest from the node.In order to extract the attribute, we first try to learn a rule that precisely extracts the node that contains the value of the attribute using a combination of strong features only.

3.1.3.1. Robust XPath generation

We are interested in generating a robust XPath query (called pXPath) that selects only the annotated nodes and none of the other nodes on annotated sample pages. An XPath is constructed using a combination of features, and our objective is to find the most general XPath (with few features) that will be robust to changes in page structure. We model robustness using two metrics: distance and support. We prefer XPaths with feature close to the annotated node over many distant features. Further, we use unannotated sample pages to define the support on unannotated pages over XPaths that do not. Giving an XPath X, we define prec(X) as the precision of the XPath, i.e. prec(X) is the ratio of the number of correct nodes selected by the XPath and the total number of nodes selected.We define dist(X) to be the maximum distance of the features of the XPath on unannotated pages.This is the fraction of unannotated pages in which the XPaths selects one or more candidates. We expect the XPaths we generate to select all annotated nodes; hence, support on annotated pages is 1.We generate an XPath that satisfies the followingproperties:

1) The XPath selects all annotated nodes, recall is 1.
2) Among all XPaths satisfying (1), precision prec(X) is maximum. 3) Among all XPaths satisfying (1) and (2) dist(X) is minimum.

4) Among all XPathssatisfying (1), (2) and (3) above, sup(X) is maximum.

We use an Apriori style algorithm to generate an XPath satisfying the above 4 properties. Let S be the set of features generated using the feature generation procedure described in the previous subsection. Algorithm 3 starts with XPath candidates C1, each of which has a single feature from S. In each subsequent iteration we combine features to generate a set of candidates Ck+1 that are more specific than the candidates Ck consideredin the previous iteration. Note that Ck contains candidate sets of features, each set with exactly k features

Algorithm 3: Learn XSLT rule

Input: Sample pages (annotated and unannotated), feature
set S,
Output: Maximum precision XPath;
C_1= {{f} : f ε S};
k=1; min dist =∞; max sup =0; max prec=0;
bestXPath = max precXPath = NULL;
while C_k /= □ **do**
L_{k+1}= □ ,
for all XPaths X(F), F in C_k**do**
if X selects all the positive nodes in P+ **then**
if X selects a negative node in P- or X matches more than one node in an un annotated page **then**
L_{k+1}= L_{k+1} U {F};
if(prec(X)> max prec) or
(prec(X) = max prec and
dist(X) < min dist) or (prec (X) = max prec
and dist(X) = min dist and
sup(X) > max sup) **then**
maxprec = prec (X);
maxprecXPath = X;
min dist = dist (X);
max sup = sup(X);
end if
else if (dist(X) < min dist) or
dist(X) = min dist and (sup(X) > max sup))
then
best XPath = X;
min dist = dist(X);
max sup = sup (X);
maxprec =1;
end if
end if
end for
for allXPAth X(F), F ε L_{k+1} do
if best XPath /= NULL and
((dist(X) > min dist) or
(dist (X) = min dist and sup (X)≤ max sup))
then
delete X from Lk+1;
end if
end for
C_{k+1} = { K+1-sets F: all K-subsets of F are in L_{k+1}};
k=k+1;
end while

ISSN: 1992-8645www.jatit.org E-ISSN: 1817-3195

We evaluate our rule learning system over many large sites from across 4 different verticals namely product, business, news etc. Let P+ be the set of annotated nodes and p- be all other nodes on annotated pages. In the K^{th} iteration, we examine candidate XPaths X corresponding to feature sets in Ck, and prune the XPaths that do not cover all the annotated nodes in p+. We also keep track of the best XPath X seen so far - this XPath satisfies properties (1) through (4). It has the minimum dist(X) from among XPaths satisfying properties (1) and (2), and the maximum support from the XPaths having distance dist (X). The best XPath is either stored in max_prec_XPath (if max_prec < 1) or best_XPath (if max_prec = 1).In order to generate the set of candidates Ck+1, we store in Lk+1 all the feature sets in Ck that can potentially lead to XPaths that are superior to the best XPath found so far. Observe that the distance of XPaths is non-decreasing and the support of XPaths in non-increasing as they become more specific in subsequent iterations. Thus, we only add to Lk+1 feature sets whose corresponding XPaths have a precision less than 1. Further, we prune from Lk+1 feature sets whose XPaths are inferior to the current best XPaths in terms of distance and support. The candidates in Ck+1 are then call sets of F containing k+1 features such that every k-subset of F belongs to Lk+1. Further, we prune from Lk+1 feature sets whose XPaths are inferior to the current best XPaths in terms of distance and support. The candidates in Ck+1 are then call sets of F containing k+1 features such that every k-subset of F belongs to Lk+1. These sites typically are script generated and hence have good template structure. The clustering on these sites gave us a total of ≈ 300 clusters.

3.2. Extraction Subsystems

The learning subsystem learns a single XSLT rules from sample of pages belonging to the cluster that are applied to the stream of crawled web pages to extract records from them.In the following subsections, we describe the algorithms employed by the key components of the extraction subsystem in detail.

3.2.1. Rule matching

For each crawled, we need to determine the matching rule and apply it to extract the record from the page. This is done in two steps. We first determine the set of matching rules for the page based on the page URL.The final rule is subsequently chosen based on the page shingle vector.To accomplish the first step, we associate a URL regex with each rule. The URL regex has a syntax similar to URLs but is allowed to contain wildcards, e.g., "http://domain-name/product/(.*)/detail.html". During learning, we select the URL regexes for a rule to be the most specific regexes that matches all the URLs contained in the cluster for the rule.To prevent over fitting, we impose certain size restrictions on the URL regexes.In the first step, we filter out rules whose URL regexes does not match the page URL. Then, in the second step, we further narrow down the rules set to those whose shingle signatures math the page's shingle vector.For a page, there can be more than one matching rule base on the URL regexes and shingle pattern. In our experiments, we have found that on an average approximately 2 rules match each page. Note that using URL regexes helps to reduce the number of possible candidate rules for shingle comparisons.

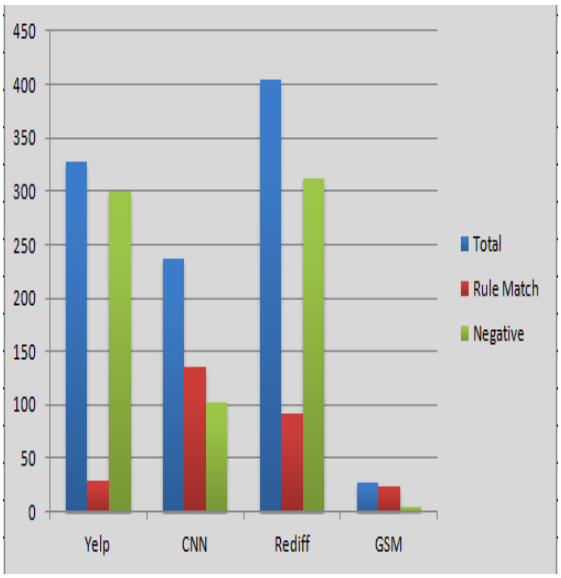

Fig 5. Rule match

Matching a URL to URL regexes is computationally much cheaper compared to computing the page shingle.

3.2.2. Rule monitoring

Web sites are dynamic with the content and structure of pages changing constantly.Examples of content changes are price changes, rating changes,etc. Page structure changes can happen due to products going on sale or out of stock, addition of reviews, variable number of ads, etc.Values ranging from 10% to 50%. Row i of the

Journal of Theoretical and Applied Information Technology
31st March 2012. Vol. 37 No.2

ISSN: 1992-8645www.jatit.org E-ISSN: 1817-3195

table contains the number of sites that changes i times. It can be seen that several sites changed multiple times over this period. In fact, one site www.amazon.com changed 4 times in one month.The number of unique sites that changed over the 30 day period is 17(40%) and the number of site changes is 27 (63%).The above coverage-based framework detects only structural changes. But a rule can fail in three different ways:Shingle changes: Due to changes in the Web page structure, the rule might not apply to previously applicable pages, thus resulting in a potential reduction of coverage. Null extraction: If the structure of the XPath pointing to a particular attribute changes with the page's shingle still conforming to the rule's shingle signature, then the rule gets applied but the rule application can result in the attribute not getting extracted anymore. Incorrect extractions: This scenario is similar to the one above except that the rule application can result in incorrect extractions. Note that this impacts extraction precision.

In order to identify rule breakage, the rule monitoring component identifies 10-20 sample URLs (called bellwether URLs) from each cluster within a site, and crawls them periodically.

3.2.3. Rule reuse

Two major findings from our rule breakage detection experiments in section 4 are:1) Even minor changes in page structure can cause pages shingles to change thereby flagging rule breakage. Approximately 2 % of web sites experience some sort of page structure change each day.2) Sites undergo partial and not complete changes. For example, for the false positives at the bottom of table 4, only a small fraction (≤ 10%) of rules breaks within each site. Hence a site change may be signalled even if only a small fraction of pages within the web site changes. Let C be anew cluster and C_1, C2, C_k be the old clusters with which Chas non-empty URL overlap. Let $U=U_{ri}(C \cap C_i)$ be the URLs in the intersection of C and the old clusters C_i. The matching cluster C_i whose rule we reuse for C is selected using two tests: a shingle test and field test. Shingle tests for detecting matching cluster pairs C, C_i have two variants. Below, they are listed in decreasing order of strictness.Cluster match: Clusters C and C_i have the same masked shingle signature.Page match: The shingles of all the pages in C match the masked shingle signature of C_i. The perception here is that if C_i and C have similar shingles then we can reuse C is rule for the new cluster C. In case of ties, we select the cluster with the largest importance score. Let C_m be the matching cluster selected by one of the above two techniques.Then R_m, the rule for C_m, is subjected to the field test. The field test employs both the old and new versions of the page for every URL in U and is as follows: We apply R_m to the old and new versions of the pages in U. For the field test to be successful, we require that,the extracted values for constant attributes should match between the old and new versions of the pages, and (2) R_m should successfully extract all the mandatory attributes from the new version of the pages. If R_m passes the field test, it is associated with C. Otherwise C is scheduled for annotation.

4. EXPERIMENT ANALYSIS

The clustering algorithm and Apriori style algorithm are implemented in .NET.Vertex extracts the index terms and stores it as records and also it works on the dynamic web pages. Figure 8 is the extraction of URLs, Figure 9 is the spitting of URLs, Figure 10 is the rule matched sites, Figure 11 represents extracting the contents, Figure 12 showing the result as attribute of records.Vertex handles end to end extraction tasks and delivers close to 100% accuracy for most attributes.

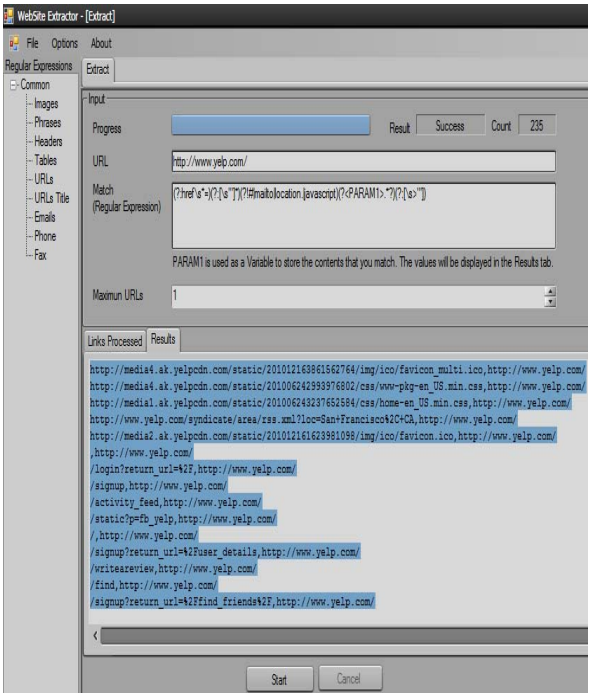

Fig 6. Extraction of URLs.

Journal of Theoretical and Applied Information Technology
31st March 2012. Vol. 37 No.2

ISSN: 1992-8645www.jatit.org E-ISSN: 1817-3195

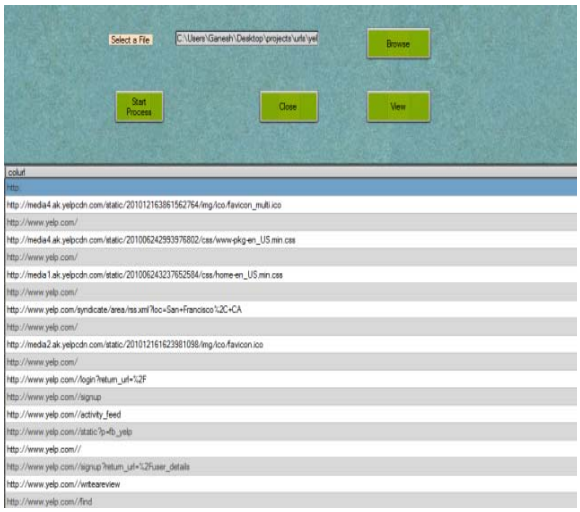

Fig 7. Spitting the URLs

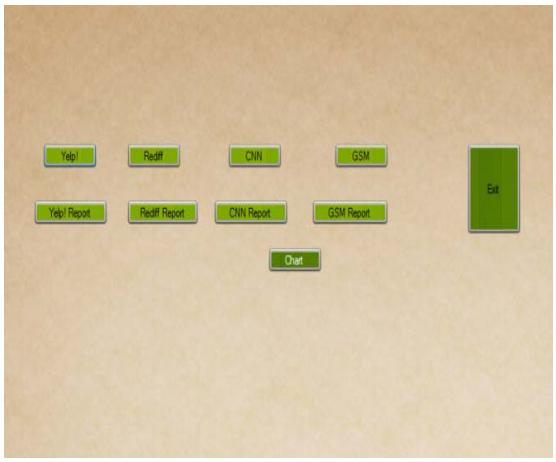

Fig 8. Rule matched sites

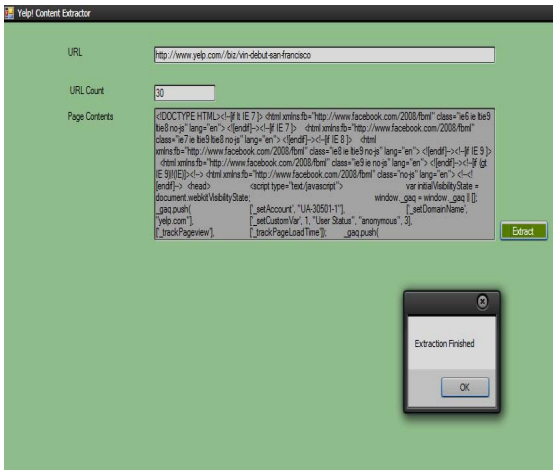

Fig 9. Extracting the contents

name	goodforgroups	AcceptsCreditCards	parking	attire	pricerange	goodforkids	takereservation	delivery
Mo's Grill	Yes	Yes	Street	Casual	$$	Yes	No	No
Precita Park Cafe		Yes		Casual		Yes		
SF Mixology	Yes	Yes	Street					
Thai Idea Vegeta...	Yes	Yes	Street	Casual	$$	Yes	Yes	No
Udupi Palace	Yes	No	Street	Casual	$$	Yes	Yes	Yes
Vin Debut	Yes	Yes	Street					
Aesthetics by Ver...		Yes	Street					
China Palace	Yes	Yes	Private Lot	Casual	$	Yes	Yes	No
Chiro-Health		Yes	Garage					
Cyclops Tattoo		No	Street			No		
El Gallo Giro Tac...	Yes	No	Street	Casual	$	Yes	No	No
EndGames Improv	Yes	Yes				Yes		
Glow Skin Care S...		Yes	Street					
Inspired Skin Care		Yes	Street					
Moth and Dagger...		No	Garage, Street			No		
Noori Pakistani &...	Yes			Casual	$	Yes	Yes	
Aesthetics by Ver...		Yes	Street					
Alisha Valverde ...		Yes	Garage					
Cyclops Tattoo		No	Street			No		
Glow Skin Care S...		Yes	Street					
I-Creation		No						
Julie Dy		No						

Fig 10. Showing the result as records

5. CONCLUSION

In this project, we described the design and performance of the vertex based information extraction platform. To work at Web scale, Vertex relies on a crowd of algorithmic innovations in Web page clustering, robust wrapper learning, detecting site changes, and rule relearning optimizations. Vertex handles end to end extraction tasks and delivers close to 100% accuracy for most attributes.Our current research focus is on reducing the editorial costs of rule learning and relearning without sacrificing accuracy. Vertex currently uses page-level shingles for clustering pages and detecting site changes. However, since structural shingles can be sensitive to minor variations in page structure, the total number of rules and rule breakages can be high. We are currently investigating XPath-based alternatives to page-level shingles to address both these issues. We are exploring ways of exploiting site -level structural constraints to boost extraction accuracy.

REFERENCES:

[1] Web-Scale Information Extraction with Vertex Pankaj Gulhane , Amit Madaan , Rupesh Mehta , Jeyashankher Ramamirtham ,Rajeev Rastogi, Sandeep Satpal , Srinivasan H Sengamedu , Ashwin Tengli , Charu Tiwari ,2011 IEEE

ISSN: 1992-8645www.jatit.org E-ISSN: 1817-3195

[2] TEXT: Automatic Template Extraction from Heterogeneous Web Pages Chulyun Kim and Kyuseok Shim, Member, IEEE Transaction on data and knowledge Engineering VOL. 23, NO. 4, APRIL 2011

[3] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, Syntactic clustering of the web. In WWW, 1997.

[4] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards automatic data extraction from large web sites. In VLDB, 2001.

[5] Y. Zhai and B. Liu. Web data extraction based on partial tree assignment. In WWW, 2005.

[6] A. Arasu and H. Garcia-Molina, "Extracting Structures Data from Web Pages," Proc. ACM SIGMOD, 2003.

[7] I.S. Dhillon, S. Mallela, and D.S. Modha, "Information-Theoretic Co-clustering," Proc. ACM SIGKDD, 2003.

[8] B. Long, Z. Zhang, and P.S. Yu, "Co-Clustering by Block Value Decomposition, "Proc. ACM SIGKDD, 2005.