Applied Intelligence
https://doi.org/10.1007/s10489-019-01499-0

DCADE: divide and conquer alignment with dynamic encoding for full page data extraction

Oviliani Yenty Yuliana[1] · **Chia-Hui Chang**[1]

Abstract

In this paper, we consider the problem of full schema induction from either multiple list pages or singleton pages with the same template. Existing approaches do not work well for this problem because they use fixed abstraction schemes that are suitable for data-rich detection, but they are not appropriate for small records and complex data found in other sections. We propose an unsupervised full schema web data extraction via Divide-and-Conquer Alignment with Dynamic Encoding (DCADE for short). We define the *Content Equivalence Class* (*CEC*) and *Typeset Equivalence Class* (*TEC*) based on leaf node content. We then combine HTML attributes (i.e., id and class) in the paths for various levels of encoding, so that the proposed algorithm can align leaf nodes by exploring patterns at various levels from specific to general. We conducted experiments on 49 real-world websites used in TEX and ExAlg. The proposed DCADE achieved a 0.962 F1 measure for non-recordset data extraction (denoted by F_D), and a 0.936 F1 measure for recordset data extraction (denoted by F_S), which outperformed other page-level web data extraction methods, i.e., DCA (F_D=0.660), TEX (F_D=0.454 and F_S=0.549), RoadRunner (F_D=0.396 and F_S=0.330), and UWIDE (F_D=0.260 and F_S=0.081).

Keywords Deep web data extraction · Divide-conquer alignment · Dynamic encoding · Full-schema induction · Multiple template pages

1 Introduction

Data extraction from the deep web is a reverse engineering task that distils a predefined template and embedded data from dynamic pages to recover the original data. Over the past decades, many data extraction and wrapper induction techniques have been proposed with varying degrees of automation, including supervised [9, 15, 22, 23], semi-supervised [2, 4, 10, 14, 20], and unsupervised (ExAlg [1], WEIR [3], RoadRunner [8], OXPath [12], Vertex [13], FivaTech [16], [21], TEX [25], STEM [30], DCA [31]) learning algorithms. Unsupervised learning algorithms are especially attractive because they can extract web data without human intervention.

✉ Chia-Hui Chang
chia@csie.ncu.edu.tw

Oviliani Yenty Yuliana
oviliani@gmail.com

[1] CSIE, National Central University, Taoyuan 32001, Taiwan

Most unsupervised approaches assume the data to be extracted are located in data-rich sections, such as search result records. Therefore, repeat pattern mining can be applied to discover recordsets from a single page through proper abstraction or encoding, such as, TEGRA [7], SYNTHIA [19], AutoRM [24], MiBAT [27], CTVS[28], DIADEM [29], DEPTA [32], Dual-TLBO [33]. However, full schema web data extraction aims to make a complete data extraction with the help of multiple pages (e.g., ExAlg [1], UWIDE [6], RoadRunner [8], FivaTech [16], TEX [25], DCA [31]). Thus, the task is not only to detect and align records in the data-rich sections, but also to align all data in the complete pages, including context-dependent advertisements and dynamic changing menus.

New challenges arise when we try to align both *list pages* and *singleton pages*. *List pages* usually contain a large list of products or records, such as search results, whereas *singleton pages* contain detailed information about an item in a page. Thus, more complex information, such as variant formats, menu bars, and small recordsets need to be aligned. For example, Fig. 1 shows two singleton pages from IMDB.com with slightly different presentation

Published online: 22 July 2019

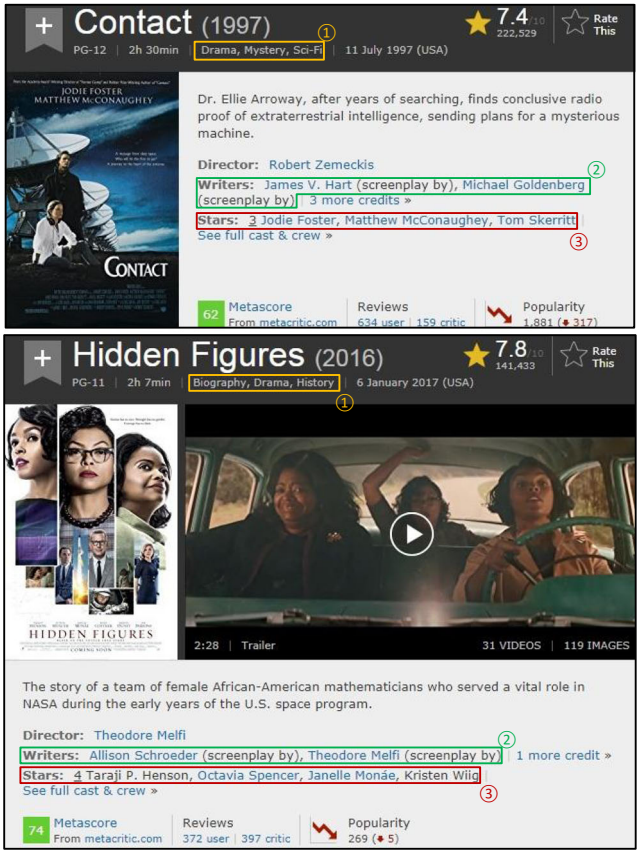


Fig. 1 An example of singleton pages (from IMDB.com) with two presentation styles

styles (with or without a preview video), small recordsets (e.g., writers or stars), a dynamic changing movie category section, and context-dependent advertisements (not shown in the clipped pages).

Among the existing approaches for full schema web data extraction, some algorithms (e.g., ExAlg, RoadRunner, and TEX) use HTML tags and words as the basic units for alignment, and some algorithms (e.g., FivaTech and WEIR) use leaf nodes from Document Object Model (DOM) trees as processing units. Whether or not word or tag tokens (small granularity) or leaf nodes (larger granularity) are used as basic units, the goal of web data extraction is to align these basic units with the same role or purpose in the same column. However, owing to the choice of different processing units, the number of output data columns and the complexity of alignment are different for different approaches. For example, RoadRunner often generates a small number of data columns and TEX usually outputs a large number of data columns. Although small granularity may be desirable for fine data extraction, excessive sparse data columns without matching components are useless. Thus, we want the system to be able to merge sparse units to the larger granularity.

Although ExAlg and TEX are full schema data extraction approaches, their evaluation only considered the selected data columns of data-rich sections. In fact, the performance of these systems across all data columns are far poorer than for the data-rich section mainly because they only align template patterns that share the same path and text string. In many cases, templates do not share the same path or text string, so many data columns cannot be correctly aligned.

Full schema data extraction from singleton webpages is more difficult and challenging than from list pages for the following reasons:

- There are more leaf nodes to align (thousands in singleton webpages compared with hundreds in list pages).
- Singleton pages may contain several small lists or sets, with optional data and similar patterns in a webpage.
- Leaf nodes with the same template role may have different paths or text contents for various reasons, such as grammar. For example, the path for text contents "Stars" in the two example pages of Fig. 1 are different due to the extra preview video clip.

The research goal of this paper is to induce a full schema from either multiple singletons or multiple list pages of the same template for data extraction. We use dynamic encoding to abstract leaf nodes for pattern discovery and design a divide-and-conquer alignment based on the prioritized encoding. We start from the most specific encoding, *Content Equivalence Class* (*CEC*), to discover template (with the same text contents) patterns. We extend this to data patterns with different contents by clustering leaf nodes with the same token types into the same *Typeset Equivalence Class* (*TEC*). With multiple levels of encoding, the proposed method could better align data nodes that have different text contents.

The remainder of this paper is organized as follows. Section 2 introduces a formal problem definition and the motivation behind the algorithm. Section 3 describes the proposed method, and Section 4 presents the performance evaluations. Section 5 provides a comparison with related work on web data extraction techniques, and Section 6 concludes the paper and proposes future work.

2 Problem definition

Given m web pages from a website, where the pages are of the same template, the problem of full schema web data extraction is to divide input pages into small pieces

of text contents such that text strings of the same role can be aligned in the same column of an $m \times n$ matrix. Note that if there exists a list or set in the input, a new matrix will be generated for each list or set. Since HTML tags are used for style presentation and readers are interested in text content, we only align text contents and we ignore HTML tags for ground truth preparation, i.e., the expected output. Based on DOM tree parsing, we align leaf nodes from different pages in the same column if they share the same role. Each column in the output has special characteristics or roles, i.e., column type. For example, columns can be mandatory (if no page contains a null value) or optional (if there are missing elements in a page). Some columns are considered as templates if they have the same text contents, while columns with varying text contents are often data columns. Therefore, each column is labeled as either *mandatory template* (*MT*), *optional template* (*OT*), *mandatory data* (*MD*), or *optional data* (*OD*). Further, repetitive columns may form a record list or set, denoted by *mandatory RecordSet* (*MR*) or *optional RecordSet* (*OR*). The final output of the data extraction system is one master aligned table with template/data/list columns, and several small aligned tables for each recordset (see Fig. 8).

3 Proposed method

The idea behind DCADE is to find trustable reference points to divide the big problem into smaller problems. As shown in Fig. 2, the algorithm consists of three phases including mandatory template detection for page segmentation, pattern discovery in segment for data-rich section detection,

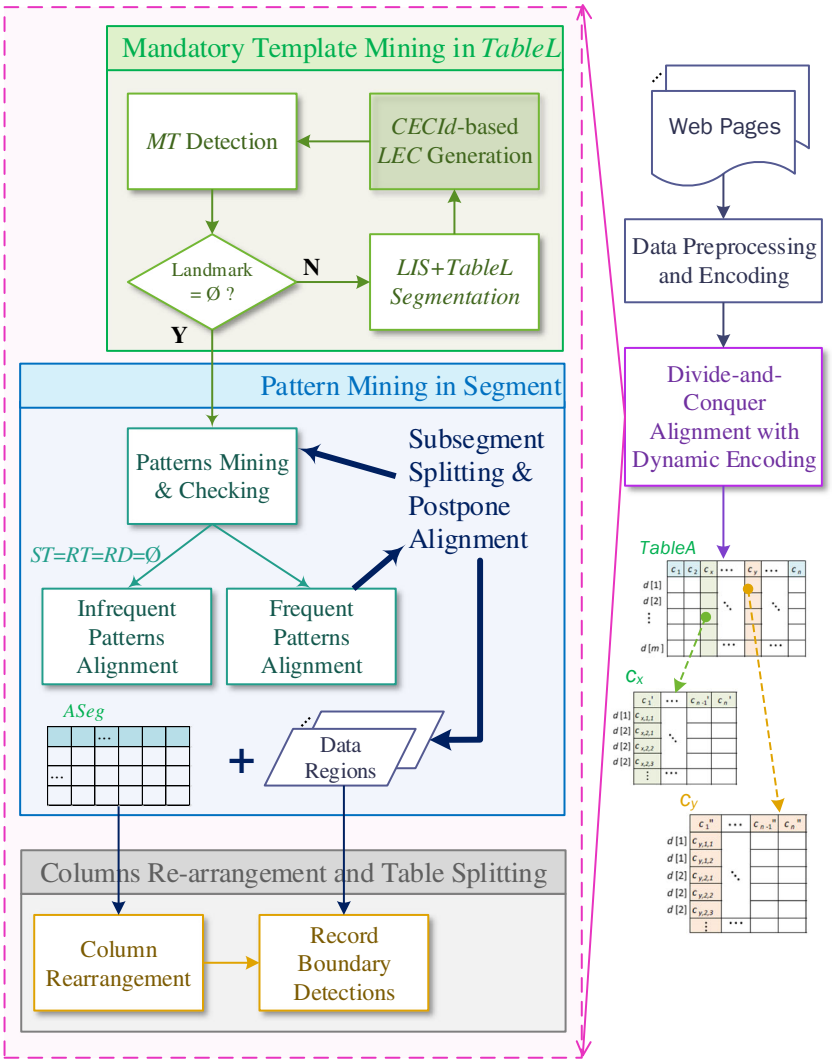


Fig. 2 System architecture of DCADE

and table splitting and column rearrangement. We first look for mandatory templates which is defined by *Content Equivalence Classes* (*CEC*) that occur only once in each sequence as landmarks and then apply *Longest Increasing Subsequence* to keep consistent landmarks for sequence segmentation.

To discover frequent patterns in each segment in the second phase, we define several encoding schemes to abstract data leaf nodes into *Typeset Equivalence Class* (*TEC*) such that leaf nodes with different text contents can share the same *TECId*s if they have the same data type, or similar path, or HTML attribute class/id sequence. Different from existing approaches for recordset discovery, we select *CEC* or *TEC* with a consistent gap as sub-landmarks, no matter their occurrence counts are the same or not. The challenging part is how to divide a segment into subsegments when some sub-landmarks are optional. In this paper, we design postpone alignment which processes subsegments from left to right such that sequences that have right sub-landmarks are aligned first. Meanwhile, sequences that are postponed will be aligned with the current result when their right sub-landmarks are encountered.

Finally, in the third phase, we detect record boundary in each repeat pattern region and split tables for each recordset. Meanwhile, disjunctive columns or low-density columns are merged to produce better alignment. To illustrate how DCADE proceeds, we use pseudo code Algorithm 1, 2, and 3 in Section 3.3 to show how different components are connected together.

3.1 Data preprocessing and encoding scheme

To process the m input pages, we first parse them into DOM trees (with CyberNeko) and arrange the leaf nodes into an $m \times l$ matrix table (called *TableL*) where l is the maximum number of leaf nodes in all pages. For each leaf node in the DOM trees, we keep five basic features including the (1) position *LeafIndex* and (2) text *Content* of the leaf node, as well as (3) *Path*, (4) *IDSeq*, and (5) *ClassSeq*, which represent HTML *tag*, *ID*, and *class* name sequence from DOM tree root to leaf node, respectively.

Based on the features, we further design four basic encoding schemes, *TypeSet/PTypeSet*, *PathId* and *SimSeqId*, to support the alignment process, where *PathId* and *SimSeqId* are generated by complete-link clustering.

Definition 1 (**TypeSet/PTypeSet Encoding**) We define token hierarchy for *TypeSet* encoding, as shown in Fig. 3. There are four main categories: address, date/ time, number, and mixture tokens. Each category is further divided into several token types. Based on the token type hierarchy, each text content is encoded as a set of token types. Formally, we define $TypeSet$ ($PTypeSet$) as a union of token types based on the lower (higher) layer token type as defined in (1).

$$\begin{aligned} TypeSet(u) &= \bigcup_{w \in u.Content} TokenType(w) \\ PTypeSet(u) &= \bigcup_{w \in u.Content} TokenPType(w) \end{aligned} \tag{1}$$

Note that function $TokenType$ and $TokenPType$ are implemented based on regular expression and returns the corresponding token type (code $5 \sim 17$ and $1 \sim 4$, respectively) as defined in Fig. 3.

Definition 2 (**PathId Encoding**) Two leaf nodes u and v share the same *PathId* if their tag sequence similarity $Sim(u.Path, v.Path)$ is greater than θ_{Path}, where $Sim(u.Path, v.Path)$ is defined in (2).

$$Sim(s_1, s_2) = \frac{|LCS(s_1, s_2)|}{max(|s_1|, |s_2|)} \tag{2}$$

where LCS denotes the *Longest Common Subsequence*, and $|x|$ returns the length of string x.

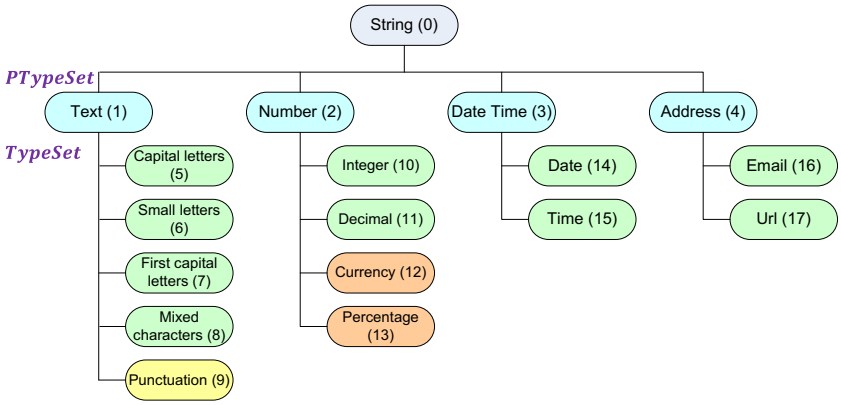
Definition 3 (**SimSeqId Encoding**) Two leaf nodes u and v share the same *SimSeqId* if their weighted *Path*, *IDSeq*, and *ClassSeq* sequence similarity $simTEC(u, v)$ is higher than θ_{TEC}, where $simTEC(u, v)$ is defined by (3) with corresponding weights ω_P, ω_I, and ω_C (0.5, 0.25, 0.25).

$$\begin{aligned} SimTEC(u,v) = {} & Sim(u.Path, v.Path) \times \omega_P + \\ & Sim(u.IDSeq, v.IDSeq) \times \omega_I + \\ & Sim(u.ClassSeq, v.ClassSeq) \times \omega_C \end{aligned} \tag{3}$$

Combining the above encoding schemes, we generate the following two types of equivalence classes.

Definition 4 (**Content Equivalence Class**) Two leaf nodes u and v are considered as a *CEC*, if $u.Content = v.Content$ and $u.PathId = v.PathId$. In other words, content equivalence class *CECId* is encoded based on the same text *Content* (denoted by the same *ContentId*) and *PathId*. In the encoding hierarchy, *CECId*, i.e. *PathId-ContentId*, is the most specific encoding, known as level 0 encoding.

Fig. 3 Token type hierarchy



Definition 5 (**TypeSet Equivalence Class**) Two leaf nodes u and v share the same *TEC*, if $u.TypeSet = v.TypeSet$ and $u.SimSeqId = v.SimSeqId$. Similar to *CEC* encoding, *TypeSet* equivalence class *TECId* is encoded based on the same *TypeSet* (denoted by the same *TypeSetId*) and *SimSeqId*. In the encoding hierarchy, the combination *SimSeqId-TypeSetId* is *level 1* encoding for *TECId*.

At the end of the data preprocessing, each leaf node in *TableL* will have two codes, i.e., *CECId* (encoding 0) and *TECId* (encoding 1). Later in the alignment process, high level codes can be generated by replacing *SimSeqId-TypeSetId* pair with *PathId* and *PTypeSetId* (or NULL) as shown in Fig. 4. In other words, the *TECId* code of leaf nodes can be replaced with a higher-level code such that leaf nodes with same role, but different content can be aligned in one column during leaf node alignment phase.

Example 1 The leading attribute "Director" in each page of Fig. 1 are fixed text *Contents* and with a similar *Path*. Thus, these leaf nodes are assigned the same *CECId* and are later used as a landmark for page segmentation. On the other hand, "Judie Foster" contains only the *FirstCapitalLetter*(7) token, while "Taraji P. Hens" has one additional token type *MixedCharacters*(8). Thus, they are assigned different *TECId*s.

3.2 Divide-and-conquer alignment

The divide-and-conquer alignment is the core basis of this research. As depicted in Fig. 2, the algorithm is composed of three phases: mandatory template mining from *CECId*, mining patterns in segments, and columns re-arrangement.

3.2.1 Mandatory template mining in *TableL*

The idea behind mandatory template mining is to find *CECId*s that occur once in every page as landmarks for page segmentation to divide the large alignment problem into sub-problems. However, since such *CECId*s may not have a consistent order across all pages, we apply the *Longest Increasing Subsequence* algorithm to the $LeafIndex$ of these leaf nodes to select consistent landmarks. Therefore, we collect leaf nodes with the same *CECId* as a *Landmark Equivalence Class* (*LEC*) and compute two attributes for each *LEC*: the *Occurrence Vector* (*OV*), which records the occurrence counts of an *LEC* in each document, and the *First Position* (*FP*) vector, which records the first occurrence position of the *LEC* in each document.[1]

As shown in the top loop of Fig. 2, we first generate the *CECId*-based *LEC* and compute the OV and FP to select candidate *MT*s (i.e., with $OV = \mathbf{1}$). We apply *Longest Increasing Subsequence* to *FP* of these candidate *MT*s in each document pair to maintain a set of consistent *MT*s (i.e., the *FP* for the selected *MT*s are consistently increased in each document) for *TableL* segmentation. The loop here means that this process is recursively called for each new segment since the OV and FP are defined with respect to each segment. The output of this *MT* mining step is an ordered sequence $MT_1, MT_2, \ldots, MT_K$. The collection of *FP*s from selected *MT*s is called *MTTable*.

3.2.2 Pattern mining in segments

In mandatory template mining, we collect leaf nodes with the same *CECId* as *LEC*s and choose *LEC*s with $OV = \mathbf{1}$ as candidate *MT*s. In addition to *LEC*s with $OV = \mathbf{1}$, there are also *LEC*s with $OV < \mathbf{1}$ or $OV > \mathbf{1}$. To differentiate frequent patterns from infrequent ones, we define support and maximum repeat (*MaxRep*) for an *LEC* e as below.

$$\begin{aligned} e.Sup &= \frac{\sum_{j=1}^{m} P(e.OV[j])}{m} \\ e.MaxRep &= Max_{j=1}^{m}\{e.OV[j]\} \end{aligned} \tag{4}$$

[1] If the rth page does not contain any leaf node with *CECId* e, we assign -1 to $e.FP[r]$.

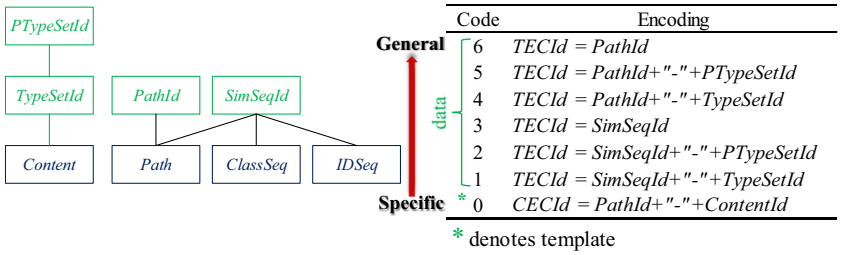


Fig. 4 Multilevel encoding schemes

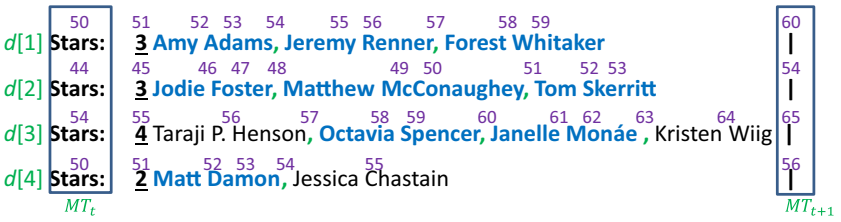
where $P(x)$ return 1 if $x > 0$ and 0 otherwise.

We call an *LEC* e frequent if $e.Sup \geq \theta_{OT}$. A frequent *LEC* e based on *CECId* is a *repeat template* (*RT*) if $e.MaxRep > \theta_{RP}$ or a *single template* (*ST*) if $e.MaxRep = 1$, where θ_{RP} is the given repeat pattern threshold. Similarly, we call an *LEC* t based on *TECId* a *repeat data* (*RD*) if $t.MaxRep > \theta_{RP}$ or a *single data* (*SD*) if $t.MaxRep = 1$. In other words, we obtain *RT* and *ST* patterns from *CECId*-based *LEC*s as well as *SD* and *RD* from *TECId*-based *LEC*s in each segment. Among these frequent patterns, *RT* and *RD* are important for recordset detection, while *SD* can be used to remove false positive *ST*.

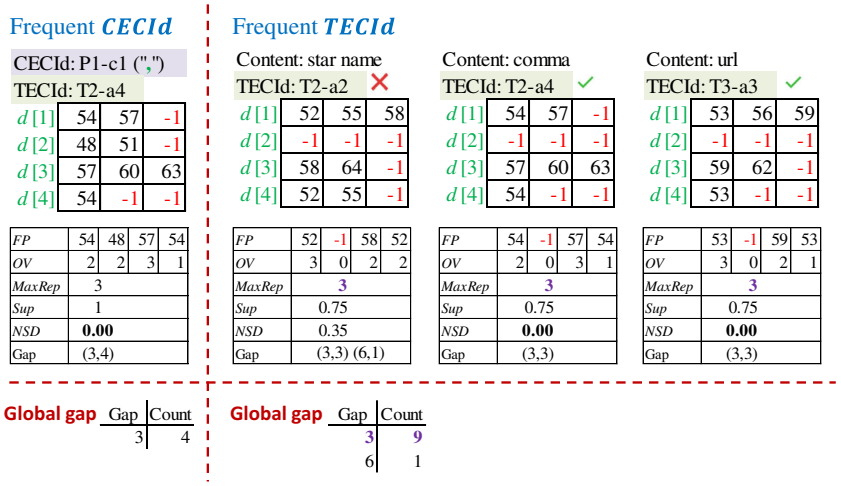
Example 2 Figure 5a shows a segment between two mandatory templates MT_t ("Stars:") and MT_{t+1} ("|") from the input page in Fig. 1. The number above each text string denotes its *LeafIndex*. For movie stars that have a hyperlink, two *LeafIndex*s are assigned: one for the text content and the other for the link. The corresponding codes, such as *PathId* and *SimSeqId*, for these leaf nodes can be found in Fig. 7.

With $\theta_{OT} = 0.3$ and $\theta_{RP} = 2$, we can find frequent *LEC* from leaf nodes with $Content =$ ", " based on their *CECId* code P1-c1 (where the first code is the *PathId* and the second code is the *ContentId*) as shown in Fig. 5(b). Since the $MaxRep$ equals 3, it is considered an *RT* pattern. Three frequent *LEC*s based on *TECId* can be found from leaf nodes with star names, commas, and URLs respectively, which we call *RD* patterns.

For each frequent *LEC*, we maintain a position matrix *PosMatrix* ($m \times MaxRep$) which keeps the $LeafIndex$ of the leaf nodes in each page. For example, the *LEC* with *TECId* code T2-a4 occurs in position 54 and 57 in $d[1]$ and is not shown in $d[2]$ because of different presentation style (see Fig. 1), i.e. the *Paths*, *IDSeqs*, and *ClassSeq*s in $d[2]$ are



(a) A recordset in one segment.



(b) Frequent *CECId* and *TECId* in (a)

Fig. 5 Repeat pattern mining example

	TECId	T1-a1	T3-a3	T2-a4	T3-a3	T2-a4	T3-a3	T2-a4	T2-a4
	CECId	P1-c9		P1-c1		P1-c1		P1-c1	P1-c3
A B A B A -	$d[1]$	50	53	54	56	57	59	-1	60
- - - - - -	$d[2]$	44	-1	-1	-1	-1	-1	-1	54
- B A B A B	$d[3]$	54	-1	57	59	60	62	63	65
A B - - - -	$d[4]$	50	53	54	-1	-1	-1	-1	56
		MT_t	LM_1	LM_2	LM_3	LM_4	LM_5	LM_6	MT_{t+1}

Note that **A** = T3-a3 and **B** = T2-a4

Fig. 6 Aligning selected *RD*s in Fig. 5b

different from the other documents, leading to different *SimSeqId*. Therefore, the second rows of these matrices are -1.

In this paper, we only consider frequent patterns with unit length and do not combine them to generate patterns with longer length as traditional methods, e.g. UWIDE [6]. This is because many heuristics that are used to prioritize patterns get weaker when too many patterns are discovered. Instead, we propose some checking mechanisms to remove false positive patterns, and consider patterns of the same type together using a trial strategy to select the best alignment result (with the smallest number of aligned columns) based on either *ST*, *RT* or *RD*. Note that the process is repeated for each subsegment as indicated in the second loop of Fig. 2, but with a priority splitting policy to decide which type of patterns are used. Again, *OV* and *FP* have to be calculated with respect to the given subsegment as mentioned above. The details of this step are described below.

– Removing False Positive *ST*s, *RT*s, and *RD*s

Since many frequent *LEC*s could be discovered, we need some mechanisms to remove duplicates between *ST*s and *SD*s and to check the consistency of *RT*s and *RD*s. For optional patterns, if an *ST* e and *SD* t have the same *TECId*, and e's occurrence vector is subsumed by that of t (i.e. $e.PosMatrix \subset t.PosMatrix$), we remove *ST* e. Owing to space limitations, we focus on consistency checking among *RT* and *RD*.

For *RT* and *RD* patterns, we use their gap statistics to remove irregular patterns and retain patterns with a consistent gap. First, we compute the average gap μ (between two adjacent *LeafIndex* values) and standard deviation σ among gaps to obtain normalized standard deviation ($NSD = \sigma/\mu$) for each *RT* (*RD*) in the same segment. We then remove *RT*s (*RD*s) with NSD greater than θ_{NSD}.[2]

Second, we sum the gap statistics for all *RT* to find the **global gap** distribution. We consider the gap with the largest count in the global gap as a **reference gap** (*RefGap*) and remove *RT*s (*RD*s) with different reference gaps. The assumption behind this heuristic is that each segment contains only one recordset, so the most frequent patterns discovered will have a consistent gap. Therefore, we keep potential frequent patterns with the consistent gap as sub-landmarks to guide the following alignment procedure.

Example 3 In the bottom of Fig. 5b, we see the local gap distribution for each *RD* as well as their global gap distribution. Most frequent *RD*s have a gap equal to 3, leading to a zero NSD. However, *TECId* T2-a2 (denoting most star names) has 1 adjacent gap equal to 6 (between *LeafIndex* 58 and 64 in $d[3]$). Since *TECId* T2-a4 and T3-a3 already have zero NSD, θ_{NSD} become zero according to footnote 2. Therefore, we remove *TECId* T2-a2.

– Aligning Frequent Patterns

To use the selected patterns (of the same pattern types) for subsegment splitting, we need to align leaf nodes of these selected patterns to divide the segment into sub-problems. Let the leaf nodes (represented by their encoding codes) from the selected patterns be ordered by their *LeafIndex*, as shown in Fig. 6. We then use the longest sequence pattern as the centroid and iteratively align other sequence patterns with the centroid via pairwise string alignment (PSA) algorithm. Here, we adopt Needleman-Wunsch algorithm [18] with the following matching score.

$$PSA(i,j) = max \begin{cases} PSA(i-1,j-1) + match(s1[i], s2[j]) \\ PSA(i-1,j) - 2 \\ PSA(i,j-1) - 2 \end{cases} \quad (5)$$

$$match(a,b) = \begin{cases} 1 & if\ a = b\ \&\ a\ is\ a\ TECId \\ 2 & if\ a = b\ \&\ a\ is\ a\ CECId \\ -4 & otherwise \end{cases} \quad (6)$$

Example 4 Let "A" and "B" denote the selected *RD* patterns with *TECId* T3-a3 and T2-a4, respectively. Thus, we represent the leaf nodes as "ABABA", "BABAB" and "AB" for $d[1]$, $d[3]$ and $d[4]$ respectively, based on the *PosMatrix*. Choosing the first string $d[1]$ as the center and iteratively

[2] If there exists frequent patterns *RT* or *RD* with $NSD = 0$, we will set θ_{NSD} to 0, otherwise $\theta_{NSD} = 0.5$.

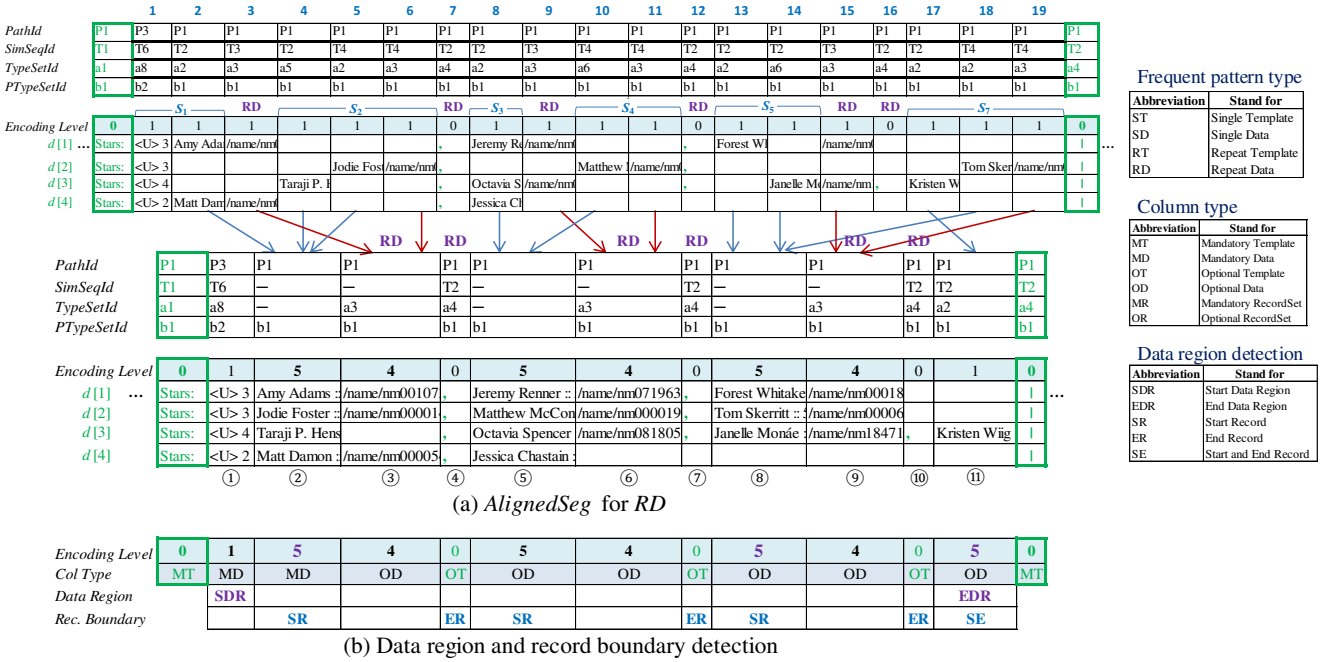


Fig. 7 Examples of **a** an aligned segment, **b** data region and record boundary detection

aligning $d[3]$ and $d[4]$ with the center, we can obtain the aligned string and *LeafIndex* matrix, as shown in Fig. 6.

We use the aligned frequent templates as sub-landmarks LM_1, LM_2, ..., LM_K to divide the segment into $K+1$ subsegments (denoted S_1, S_2, ..., S_{K+1}). The pattern mining procedure is then recursively applied to each subsegment as indicated in the second loop of Fig. 2. However, since there might be optional sub-landmarks (indicated by -1 in Fig. 6) for some documents, we need to exclude them from the template/data pattern mining and postpone the processing of these documents until later subsegments when the right sub-landmark is not optional ($\neq -1$).

Example 5 As shown in Fig. 6, there are seven subsegments to be processed recursively. The -1 values for $d[2]$ and $d[3]$ in the first column indicates that only $d[1]$ and $d[4]$ would join the template/data mining process in the first subsegment. $d[3]$ would be aligned with the aligned columns at the end of the 2nd subsegment mining procedure. Similarly, $d[4]$ would be ignored from the pattern mining procedure until the last subsegment, where we align it with other aligned columns.

– Infrequent Patterns Alignment

The recursive pattern mining procedure stops if no frequent patterns are discovered in a subsegment, where we can align all leaf nodes that could not become frequent patterns in this subsegment directly by multiple string alignment based on the center-star algorithm. This is like a terminal condition in the recursive call. The output of the recursive procedure is the aligned table. Thus, the final step of the second phase is to align postponed documents with the aligned table.

Example 6 As shown in the top two tables of Fig. 7a, the aligned frequent RD patterns divide the segment into seven subsegments $(S_1, \cdots, S_7)$. According to column LM_1 of Fig. 6, we align *LeafIndex* before 53 in $d[1]$ and $d[4]$ to produce the first subsegment S_1. There are no leaf nodes to be aligned for the second subsegment S_2. However, since the $d[3]$ has a value of 57 in column LM_2 of Fig. 6, we align *LeafIndex* before 57 with the first three aligned columns. Since the text node "Taraji P. Henson" (*LeafIndex* 56) in $d[3]$ (*TECId* code T2-a5) cannot be aligned with the second column (*TECId* code T2-a2), a new column (4th) is inserted. The process continues until all seven subsegments are aligned. Then, $d[2]$ (all leaf nodes) and $d[4]$ (leaf nodes after 54) are aligned.

In summary, if new sub-landmarks are discovered for a segment, we call the subsegment pattern mining procedure recursively for each subsegment S_k (k=1 to $K+1$) and append the new aligned results with those of previous subsegments. During subsegment pattern mining, documents that do not contain the frequent pattern L_k are marked as postponed. In the terminal procedure of each

recursive call, we align all leaf nodes in a segment, and further align the postponed documents if the the sublandmark is not -1.

Note that for each segment (between two *MT*s), we may find *RT*, *RD* and *ST* patterns in one segment at the same time. If we align all frequent patterns at once, we may break the repeat pattern of recordsets. Therefore, we choose to align frequent patterns of the same type simultaneously and select the pattern type with the minimum aligned columns.

3.2.3 Columns re-arrangement and table splitting

Although the strategy of aligning frequent patterns followed by infrequent patterns can put most leaf nodes in proper columns, data nodes could not be aligned well because of different *TECId* codes. Therefore, we conduct column re-arrangement to merge disjunctive columns via dynamic encoding and merge low-density data columns to reduce extra columns. Meanwhile, if a subsegment is a data region, we further conduct record boundary detection for table splitting.

– Merging Disjunctive Columns

Because the DCADE algorithm works on two specific encoding levels (0 and 1) for landmark pattern mining, many data leaf nodes cannot be merged. Thus, many extra columns are inserted because of different *TECId* code. To solve this problem, we need to adjust the encoding scheme to align data nodes with higher encoding levels for common *TECId*. In this paper, we take a greedy approach to merge columns that have support (see (4)) less than one with other disjunctive columns from the left to the right, and stop when the support becomes one.

Example 7 We start from the first column of the top two tables in Fig. 7a and find the 2nd column has support 0.5. First, we merge the 2nd and 4th columns into one column with encoding level 2 to generate the common code T2-b1 for $d[1]$, $d[3]$, and $d[4]$. Next, the 5th column is merged with encoding level 5 to generate the common code P1-b1 for all four documents. Moving to the next column, the URL columns at the 3rd and 6th columns can be merged with encoding level 4 to generate the common *TECId* P1-a3.

– Merging Low Density Data Columns

In addition to the merging of disjunctive columns, we consider another strategy to merge low density columns between two sub-landmarks $LM_{k-1} \sim LM_k$ if their $Density(S_k)$, i.e., the percentage of nonnull leaf nodes, is less than or equal to θ_{Den} (=0.7).

$$Density(S_k) = \frac{\sum_r LeafNode(S_k, r)}{\max_r LeafNode(S_k, r) \times m} \tag{7}$$

where $LeafNode(S_k, r)$ denotes the number of leaf nodes in $d[r]$ for segment S_k.

– Record Boundary Detection

If a segment is aligned based on *RT* or *RD* patterns, it means a data region is discovered in this segment. Since there could be non-recordset data in the same segment, we need to decide the start and end position for the data region and each record boundary. Let p_1 and p_K respectively denote the column index of LM_1 and LM_K in the aligned results *TableA*. For example, $p_1 = ③$ and $p_K = ⑩$. The potential *Start Data Region* (*SDR*) and *End Data Region* (*EDR*) can be bounded by

$$\begin{aligned} SDR &= \max\{p_1 - RefGap + 1, 1\} \\ EDR &= \min\{p_K + RefGap - 1, p_{K+1} - 1\} \end{aligned} \tag{8}$$

Assume the first *Start Record* (SR_1) ranges from *SDR* to *EDR* $-2*$ *RefGap* (since there are at least two records) and the second record follows immediately after the first one. We can then compute the matching score of the first two records and choose SR_1 with the highest matching score as the starting position. Thus, the start of the second record will be $SR_2 = SR_1 + RefGap$.

$$Score(SR_1) = \frac{\sum_{i=0}^{RefGap-1} match(SR_1 + i, SR_2 + i)}{RefGap} \tag{9}$$

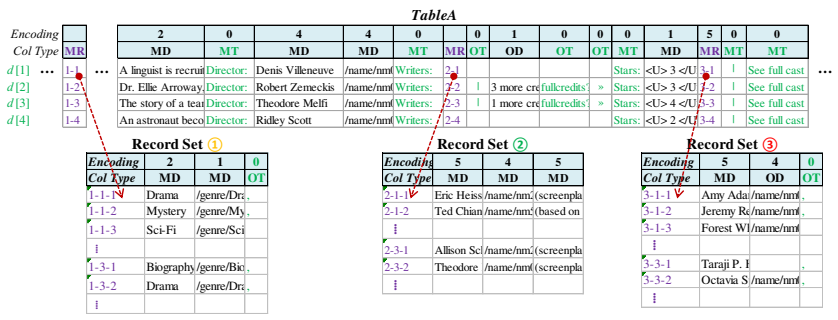


Fig. 8 Final output with split tables

We use a loop to compute the matching score for each SR_1 from SDR to $EDR - 2 * RefGap$ and break the loop if the score equals one. Finally, we mark data records with *Start Record* (*SR*) starting from SR_1 and *End Record* (*ER*) or *SE* if a record only has a field.

Example 8 Consider the aligned segment, which has $RefGap = 3$ as shown in the last table in Fig. 7a. Since $p_1 = ③$ and $p_K = ⑩$, we have $SDR = 1$ and $EDR = 11$.

To decide the start of the first record, we consider SR_1 from 1 to 5. If $SR_1 = 1$, the score is 2/3 since there are two matching column pairs that have the same *TECID* code: (②, ⑤) and (③, ⑥). If $SR_1 = 2$, there will be one additional matching column pairs: (④, ⑦). The score is 1, and we stop the loop. Therefore, we mark ②, ⑤, and ⑧ as the start of a record (*SR*). Finally, we check the remaining columns before *EDR* with the found record schema P1-b1, P1-a3, and *CECId* for encoding level 5, 4, 0, respectively. Figure 7b shows the final record boundary.

Finally, we split each data region into a separate table as follows. We keep an index in $TableA$ to represent a split data region and a pointer for each document to the corresponding row in the created table. The pointer follows the format: *SetId*−*DocId*−*RecId*. Figure 8 shows an example output with three split recordsets.

3.3 Summary

We summarize the main algorithm in Algorithm 1. First, the *DataProcessEncode* module parses the input to generate a leaf node sequence for *TableL*, and we call *MTMining* to find mandatory patterns *MTTable* for segmentation (lines 2 and 3). Then, DCADE processes each segment sequentially to generate aligned columns (lines 4 to 12). If a segment contains only one leaf node on each page, we call *AlignInFreqPattern* and *ReArrangement* directly. Otherwise, we call *MiningSegment* to generate *AlignedSeg* and concatenate it with the previously aligned *TableA* (line 10).

In Algorithm 2, the *MiningSegment* module first computes frequent $CECId$ for STs and RTs (line 1) and frequent $TECId$ for SDs and RDs (line 2) from the given segment between $u = MT_t$ and $v = MT_{t+1}$. The *MiningSegment* module then removes false positive STs via SDs (line 3) and keeps consistent RTs and RDs based on $GapRT$ and $GapRD$, respectively, via the global gap statistics (line 4). If none of the three pattern types are discovered, we align data nodes with *AlignInFreqPattern* and *ReArrangegment* (lines 7 and 8). Otherwise, we attempt to align frequent patterns based on an ST, RT, or RD pattern type (lines 13 to 19) and repeat a similar procedure ($SplittingSubSegment$) for each new subsegments (line 17) to generate data regions and augment $ASeg_p$. From the three possible pattern types, we select the split strategy with the minimum output data columns (line 20) and conduct record boundary detection for each data region in $RecDetection$ (line 21).

Algorithm 1 DCADE (Webpages).

```
1  TableA ← ∅
2  TableL ← DataProcessEncode(Webpages)
3  MTTable ← MTMining(TableL)
4  for t ← 0, |MTTable| − 1 do
5      if |MT_{t+1} − MT_t| = 2 then
6          AlignedSeg ← AlignInFreqPattern(MT_t, MT_{t+1})
7          ReArrangement(AlignedSeg)
8      else if |MT_{t+1} − MT_t| > 2 then
9          AlignedSeg ← MiningSegment(MT_t, MT_{t+1})
10     TableA ← TableA ⊕ AlignedSeg
11     TableA ← TableA ⊕ MT_{t+1}
12 end
13 return TableA
```

Note that MT_0 and $MT_{|MTTable|}$ are dummies MTs.
TableL is a global matrix. Underline denotes call by reference.

Algorithm 2 MiningSegment (u, v).

```
1  ST, RT ← MineTemplatePattern(u, v)
2  SD, RD ← MineDataPattern(u, v)
3  RemoveFalseST(SD, ST)
4  Gap^RT, Gap^RD ← RemoveFalseRTRD(RT, RD)
5  Gap^ST ←0
6  if RT = RD = ST = ∅ then
7      ASeg ← AlignInFreqPattern(u, v)
8      ReArrangement(ASeg)
9      return ASeg
10 else
11     Postpone ← ∅ // global variable
12     S_L ← u // global variable
13     for each pattern type t ∈ ("ST", "RT", "RD") do
14         ASeg^t ← ∅
15         DataRegSet^t ← ∅
16         LM^t ← AlignFrequentPattern(t) ⊕ v
17         SplittingSubSegment(t, Gap^t, LM^t, DataRegSet^t,
           ASeg^t)
18         ReArrangement(ASeg^t)
19     end
20     t* ← arg min_t(|ASeg^t|)
21     RecDetection(v, DataRegSet^{t*}, ASeg^{t*})
22     return ASeg^{t*}
23 end
```

Postpone is a global set for storing documents not processed.

Algorithm 3 SplittingSubSegment(p, Gap, LM, $DataRegSet$, $Aseg$).

```
1  if Gap ≠ 0 then
2  |   DataRegSet ←
   |   DataRegSet ∪ {p, Gap, S_L, LM_{|LM|−1}}
3  end
4  for c ← 1, |LM| − 1 do
5  |   for r ← 1, |D| do
6  |   |   if LM_c[r] = −1 then
7  |   |   |   Postpone ← Postpone ∪ {r}
8  |   |   end
9  |   end
10 |   if |LM_c − S_L| = 1 then
11 |   |   // no leaf nodes between S_L and LM_c
12 |   |   Aseg ← Aseg ⊕ LM_c
13 |   else
14 |   |   ST, RT, RT_NSD ←
   |   |   MineTemplatePattern(S_L, LM_c, Postpone)
15 |   |   SD, RD, RD_NSD ←
   |   |   MineDataPattern(S_L, LM_c, Postpone)
16 |   |   RemoveFalseST(SD, ST)
17 |   |   GapRT, GapRD ←
   |   |   RemoveFalseRTRD(RT, RD)
18 |   |   if GapRT ≠ 0 and RT_NSD = 0 then
19 |   |   |   SubLM ← AlignFrequentPattern(RT)
20 |   |   |   SplittingSubSegment("RT", GapRT,
   |   |   |   SubLM, DataRegSet, ASeg)
21 |   |   else if GapRD ≠ 0 and RD_NSD = 0 then
22 |   |   |   SubLM ← AlignFrequentPattern(RD)
23 |   |   |   SplittingSubSegment("RD", GapRD,
   |   |   |   SubLM, DataRegSet, ASeg)
24 |   |   else if ST ≠ ∅ then
25 |   |   |   SubLM ← AlignFrequentPattern(ST)
26 |   |   |   SplittingSubSegment("ST", 0,
   |   |   |   SubLM, DataRegSet, ASeg)
27 |   |   else if RT ≠ ∅ then
28 |   |   |   SubLM ← AlignFrequentPattern(RT)
29 |   |   |   SplittingSubSegment("RT", GapRT,
   |   |   |   SubLM, DataRegSet, ASeg)
30 |   |   else if RD ≠ ∅ then
31 |   |   |   SubLM ← AlignFrequentPattern(RD)
32 |   |   |   SplittingSubSegment("RD", GapRD,
   |   |   |   SubLM, DataRegSet, ASeg)
33 |   |   else
34 |   |   |   ASeg ← ASeg⊕
   |   |   |   AlignInFreqPattern(S_L, LM_c)
35 |   |   end
36 |   end
37 |   for r ← |D|, 1 do
38 |   |   if LM_c[r] ≠ −1 & r ∈ Postpone then
39 |   |   |   AligningPendingDoc(ASeg, r, S_L[r], LM_c[r])
40 |   |   |   Postpone.remove(r)
41 |   |   end
42 |   end
43 |   for r ← 1, |D| do
44 |   |   if LM_c[r] ≠ −1 then
45 |   |   |   S_L[r] ← LM_c[r]
46 |   |   end
47 |   end
48 end
```

Since the aligned frequent patterns LM^t may contain optional landmarks (line 16 in Algorithm 2), we must exclude $d[r]$ from subsegment mining if there is no right landmark (-1). Therefore, Algorithm 3 saves $d[r]$ to a $Postpone$ set (lines 5 to 9) before the recursive call to the main procedure (lines 10 to 36). However, if a documenet $d[r]$ has previously been placed in the $Postpone$ set, but the right boundary is not optional in this subsegment, i.e., $LM_c[r] \neq -1$ & $r \in Postpone$ (line 38), we call $AligningPendingDoc$ to align $d[r]$ and remove it from the set (lines 39 and 40). Thus, we can ensure that documents with optional right boundaries are processed properly through $SplittingSubSegment$.

For the left boundary of a segment, we use $S_L = u$ (line 12 in Algorithm 2) to ensure the starting position of the first $SplittingSubSegment$. We gradually update S_L (lines 43 to 47 in Algorithm 3) to the next landmark if $LM_c[r]$ is not optional. Thus, S_L keeps the left *LeafIndex* of all documents for the current segment.

Compare Algorithm 2 and 3, the main steps for each subsegment are similar, i.e., (template/data) patterns mining, frequent/infrequent patterns alignment, and subsegment splitting. The major difference is that we try for every pattern types in $MiningSegment$, but choose only one of the ST, RT, or RD patterns in $SplittingSubSegment$, i.e., we give priority of $RT > RD > ST$ if $NSD = 0$ and the priority of $ST > RT > RD$ when the NSD is not zero and $ST \neq \varnothing$ (lines 18 to 32 in Algorithm 3). If $SplittingSubSegment$ is called with frequent RT or RD patterns, we add a region with the left and right boundary from S_L and $LM_{|LM|-1}$ into *data region sets* ($DataRegSet$). Finally, we call $RecDetection$ to detect the record boundary for each data region at the end of Algorithm 2 (line 21).

4 Experimental results

We use two datasets for the experiments, including nine websites from ExAlg (242 web pages) and 40 websites in eight categories from TEX[3] (1,202 web pages). Table 1 shows the statistics about the number of web pages and the maximum number of leaf nodes ($MaxLN$) per webpage for each website, which corresponds to the number of rows and columns of input matrix *TableL*. Table 1 also shows the ground truth of aligned page-level schema for each website including non-recordset region *TableA* and data-rich section, i.e. recordsets. The ground truth follows the output data structure of RoadRunner (RR), which includes lists, tuples and basic types. For our ground truth, we ignore tuples as they could be defined differently and other approaches do not provide such information. Meanwhile, we merge leaf nodes belonging to the same comments to

[3] http://www.tdg-seville.info/Hassan/TEX

Table 1 Data and ground truth description

Category	ID	Website	*TableL* (input)			*TableA*				Recordsets				
			#Page	*MaxLN*	Density	#Col	#Template	#Data	Density	#Set	#Col	#Template	#Data	*MaxRow*
ExAlg	E01	Amazon Cars	21	206	0.817	141	135	5	1.000	1	11	3	8	54
	E02	UEFA-Players	20	99	0.850	36	28	7	1.000	1	2	0	2	501
	E03	Amazon-Pop-Artist	19	1,281	0.524	229	223	5	1.000	1	3	1	2	2,807
	E04	UEFA-Teams	20	56	0.999	56	39	17	0.997	0	–	–	–	–
	E05	www.ausopen.com	32	285	0.717	142	106	36	0.938	0	–	–	–	–
	E06	www.ebay.com	50	277	0.801	141	102	39	0.889	0	–	–	–	–
	E07	MLB Players	10	874	0.721	356	352	3	1.000	1	5	0	5	552
	E08	www.netflix.com	50	567	0.883	390	317	73	0.957	0	–	–	–	–
	E09	RPM Packages	20	6,052	0.090	24	20	4	1.000	0	–	–	–	–
Books	T01	www.abebooks.com	30	404	0.857	303	255	46	0.890	2	12	2	10	96
	T02	www.awesomebooks.com	30	351	0.708	216	191	24	0.844	1	10	2	8	47
	T03	www.betterworldbooks.com	30	715	0.892	365	300	63	0.949	2	15	7	8	478
	T04	www.manybooks.net	30	897	0.383	146	114	31	0.853	1	3	1	2	85
	T05	www.waterstones.com	30	434	0.896	301	287	13	0.860	1	3	0	3	39
Cars	T06	www.autotrader.com	30	699	0.888	496	319	175	0.909	2	4	2	2	207
	T07	www.carmax.com	30	466	0.960	391	321	69	0.957	1	9	1	8	162
	T08	www.carzone.ie	30	222	0.964	196	145	50	0.877	1	3	1	2	234
	T09	www.classiccarsforsale.co.uk	30	726	0.876	580	517	63	0.932	0	–	–	–	–
	T10	www.internetautoguide.com	30	389	0.935	309	228	81	0.979	0	–	–	–	–
Events	T11	events.linkedin.com	30	293	0.473	82	52	27	0.889	3	7	1	6	302
	T12	www.allconferences.com	30	283	0.792	115	88	26	0.900	1	3	1	2	842
	T13	www.mbendi.com	30	50	1.000	50	37	13	1.000	0	–	–	–	–
	T14	www.netlib.org	30	10	0.500	1	0	1	0.974	0	–	–	–	–
	T15	www.rdlearning.org.uk	30	136	0.502	46	39	7	1.000	0	–	–	–	–
Doctors	T16	doctor.webmd.com	31	605	0.934	551	533	17	0.849	1	3	0	3	115
	T17	extapps.ama-assn.org	30	312	0.917	270	255	14	0.922	1	3	1	2	76
	T18	www.dentists.com	30	135	0.754	61	51	10	0.960	0	–	–	–	–
	T19	www.drscore.com	30	143	0.923	122	95	25	0.926	2	4	2	2	76
	T20	www.steadyhealth.com	30	430	0.995	332	304	27	0.956	1	4	0	4	600
Jobs	T21	careers.insightintodiversity.com	30	253	0.880	175	165	10	1.000	0	–	–	–	–
	T22	www.4jobs.com	30	421	0.811	189	161	24	0.905	4	21	3	18	208
	T23	www.6figurejobs.com	30	817	0.876	652	613	37	0.974	2	4	2	2	124
	T24	www.careerbuilder.com	30	450	0.639	221	189	32	0.979	0	–	–	–	–

Table 1 (continued)

Category	ID	Website	*TableL* (input)			*TableA*				Recordsets				
			#Page	*MaxLN*	Density	#Col	#Template	#Data	Density	#Set	#Col	#Template	#Data	*MaxRow*
	T25	www.jobofmine.com	30	315	0.519	98	85	13	0.992	0	–	–	–	–
Movies	T26	www.albaniam.com	30	91	0.958	93	75	18	0.674	0	–	–	–	–
	T27	www.allmovie.com	30	643	0.547	145	117	22	0.952	6	13	1	12	404
	T28	www.citwf.com	30	129	0.857	79	61	16	0.835	2	5	2	3	241
	T29	www.disneymovieslist.com	30	202	0.781	132	105	27	0.985	0	–	–	–	–
	T30	www.imdb.com	30	1,272	0.950	1,089	733	351	0.827	5	17	3	14	350
	T31	www.soulfilms.com	30	404	0.819	259	229	29	0.937	1	2	0	2	535
Estate	T32	realestate.yahoo.com	30	748	0.852	449	354	85	0.889	10	31	1	30	407
	T33	www.haart.co.uk	30	343	0.814	228	181	47	0.930	0	–	–	–	–
	T34	www.homes.com	30	484	0.692	302	209	92	0.648	1	3	1	2	117
	T35	www.remax.com	30	366	0.883	271	207	59	0.904	5	11	5	6	290
	T36	www.trulia.com	30	1,495	0.896	954	567	383	0.953	4	17	1	16	150
Sports	T37	baseball.playerprofiles.com	30	488	0.525	53	35	18	0.972	0	–	–	–	–
	T38	en.uefa.com	30	488	0.525	53	35	18	0.972	0	–	–	–	–
	T39	www.atpworldtour.com	30	939	0.917	814	743	70	0.920	1	4	1	3	59
	T40	www.nfl.com	31	941	0.845	651	599	52	0.986	0	–	–	–	–
Average			30	599	0.776	278	228	49	0.928	2	8	2	7	351

Fig. 9 Incomplete extraction from TEX

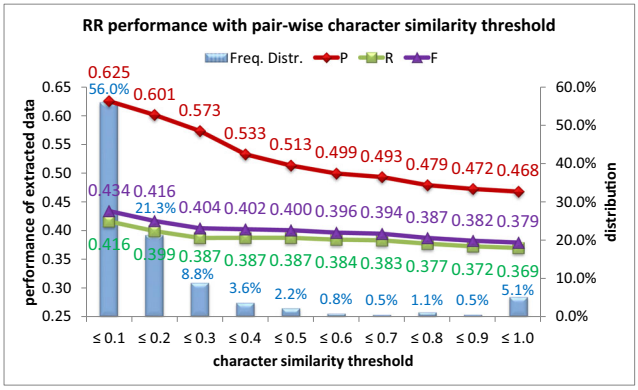
Ground Truth	TEX
9,384,100.42	384,100.42
9,384,095.07	384,095.07
9,384,066.42	384,066.42
9,384,059.73	384,059.73
9,381,500.65	381,500.65

Ground Truth	TEX
	1
1 person liked this.	1
1 person liked this.	2
	7
2 people liked this.	
7 people liked this.	

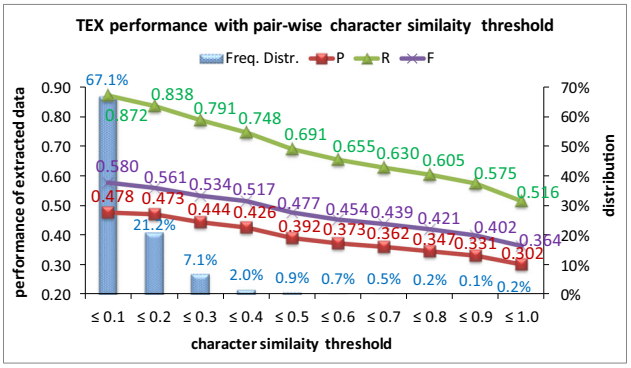
produce compact data columns, i.e. increasing the density from 0.776 to 0.928 as shown in the density columns of *TableL* and *TableA*. On average, we reduce 599 $MaxLN$ to 278 columns in the aligned output matrix *TableA*, which are further labeled as template (228) or data (49) columns per website.

For lists, we output them in separate tables and count the number of recordsets for each website. The total number of aligned columns in all recordsets and the number of template/data columns as well as the maximum number of rows per recordset are shown in Recordset part of Table 1. Among 49 websites, only five are list-pages; however, many (24) websites contain small recordsets.

Given the ground truth and the output from any approach, we can evaluate the extraction performance in terms of data columns. We use (10) to compute precision and recall for each extracted data column. We then average precisions of extracted data columns and recall of ground truth data columns as shown in (11), where *Ext* and *GT* denote the extracted and ground truth data columns respectively.



(a)



(b)

Fig. 10 Performance tuning with respect to similarity threshold for RoadRunner **a** and TEX **b**

$$
\begin{aligned}
P_c &= \frac{\#correct\ extracted\ cells\ in\ column\ c}{\#extracted\ cells\ in\ column\ c} \\
R_c &= \frac{\#correct\ extracted\ cells\ in\ column\ c}{\#columns\ in\ ground\ truth\ column}
\end{aligned} \tag{10}
$$

$$
\begin{aligned}
\bar{P} &= \frac{\sum_{c=1}^{|Ext\ column|} P_c}{|Ext\ column|},\ \bar{R} = \frac{\sum_{c=1}^{|GT\ column|} R_c}{|GT\ column|}, \\
F &= \frac{2 \times \bar{P} \times \bar{R}}{\bar{P} + \bar{R}}
\end{aligned} \tag{11}
$$

4.1 Baselines

In this work, we compare the performances of DCADE with four state-of-the-arts page-level extraction systems, including RoadRunner (2005), TEX (2013), UWIDE (2016), and DCA (2018). Since the processing units for different approaches are different, we explain how various methods are processed for a fair comparison. For example, TEX may mix links/images and text in the same file (i.e. column), therefore we help split them into several files and clean the tags related with links/images for evaluation. Similarly, UWIDE extracts HTML script tags which are unique to other approaches. Therefore, we removed the scripts from the extracted data.

Table 2 Performance comparison on data columns of *TableA*

Category	ID	Website	DCADE			DCA			UWIDE			TEX			RoadRunner		
			P_D	R_D	F_D	P_D	R_D	F_D	P_D	R_D	F_D	P_D	R_D	F_D	P_D	R_D	F_D
ExAlg	E01	Amazon Cars	**1.000**	**1.000**	**1.000**	0.333	0.800	0.470	0.145	0.667	0.238	0.158	0.600	0.250	**1.000**	**1.000**	**1.000**
	E02	UEFA-Players	**1.000**	**1.000**	**1.000**	0.292	**1.000**	0.452	**1.000**	0.143	0.250	0.318	**1.000**	0.483	0.778	**1.000**	0.875
	E03	Amazon-Pop-Artist	**1.000**	**1.000**	**1.000**	0.030	**1.000**	0.058	0.010	0.600	0.020	0.359	0.789	0.493	**1.000**	**1.000**	**1.000**
	E04	UEFA-Teams	**1.000**	**1.000**	**1.000**	0.994	0.988	0.991	0.846	0.644	0.731	0.777	0.926	0.845	0.842	0.941	0.889
	E05	www.ausopen.com	**1.000**	**1.000**	**1.000**	0.804	0.922	0.859	0.650	0.181	0.283	0.361	0.591	0.448	**1.000**	**1.000**	**1.000**
	E06	www.ebay.com	**1.000**	**1.000**	**1.000**	0.590	0.787	0.674	0.190	0.471	0.271	0.203	0.507	0.290	0.020	0.001	0.002
	E07	MLB Players	0.235	**1.000**	0.381	0.125	**1.000**	0.222	**1.000**	**1.000**	**1.000**	0.033	0.600	0.063	0.333	0.333	0.333
	E08	www.netflix.com	**0.986**	**0.997**	**0.992**	0.851	0.921	0.885	0.192	0.231	0.210	0.375	0.666	0.480	0.000	0.000	0.000
	E09	RPM Packages	**1.000**	**1.000**	**1.000**	0.500	**1.000**	0.667	0.000	0.017	0.000	0.159	0.688	0.258	0.000	0.000	0.000
Books	T01	www.abebooks.com	**0.961**	**0.958**	**0.960**	0.828	0.905	0.865	0.238	0.443	0.310	0.559	0.599	0.578	0.000	0.000	0.000
	T02	www.awesomebooks.com	**1.000**	**1.000**	**1.000**	0.819	0.817	0.818	0.175	0.503	0.260	0.321	0.411	0.361	0.913	0.875	0.894
	T03	www.betterworldbooks.com	**0.915**	**0.970**	**0.942**	0.875	0.920	0.897	0.000	0.000	0.000	0.233	0.752	0.356	0.820	0.065	0.121
	T04	www.manybooks.net	**0.991**	**0.999**	**0.995**	0.558	0.655	0.603	0.055	0.377	0.096	0.329	0.446	0.379	0.000	0.000	0.000
	T05	www.waterstones.com	**1.000**	**1.000**	**1.000**	0.482	0.800	0.602	0.085	0.597	0.149	0.052	0.518	0.094	0.967	0.372	0.537
Cars	T06	www.autotrader.com	**0.935**	**0.972**	**0.953**	0.745	0.748	0.746	0.324	0.555	0.409	0.494	0.499	0.496	0.000	0.000	0.000
	T07	www.carmax.com	**0.997**	**0.994**	**0.995**	0.450	0.872	0.594	0.338	0.456	0.388	0.268	0.498	0.348	0.500	0.101	0.169
	T08	www.carzone.ie	**1.000**	**1.000**	**1.000**	0.645	0.773	0.703	0.757	0.575	0.654	0.569	0.656	0.610	0.000	0.000	0.000
	T09	www.classiccarsforsale.co.uk	**0.968**	**0.959**	**0.963**	0.859	0.895	0.877	0.336	0.567	0.422	0.409	0.766	0.534	0.000	0.000	0.000
	T10	www.internetautoguide.com	**0.987**	**0.999**	**0.993**	0.540	0.835	0.656	0.307	0.435	0.360	0.487	0.712	0.578	0.000	0.000	0.000
Events	T11	events.linkedin.com	0.997	**0.997**	**0.997**	0.636	0.717	0.674	0.747	0.133	0.226	0.460	0.556	0.504	**1.000**	0.963	0.981
	T12	www.allconferences.com	**1.000**	**1.000**	**1.000**	0.581	0.956	0.723	0.128	0.151	0.139	0.591	0.726	0.652	**1.000**	**1.000**	**1.000**
	T13	www.mbendi.com	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.933	0.144	0.249	0.667	0.769	0.714	**1.000**	**1.000**	**1.000**
	T14	www.netlib.org	**1.000**	**1.000**	**1.000**	0.483	0.933	0.636	0.000	0.000	0.000	0.001	0.033	0.001	**1.000**	**1.000**	**1.000**
	T15	www.rdlearning.org.uk	**1.000**	**1.000**	**1.000**	0.292	0.652	0.403	**1.000**	0.429	0.600	0.350	0.805	0.488	0.883	0.252	0.393
Doctors	T16	doctor.webmd.com	0.773	**0.944**	**0.850**	**0.844**	0.835	0.839	0.280	0.751	0.408	0.427	0.607	0.502	0.667	0.118	0.200
	T17	extapps.ama-assn.org	**1.000**	**1.000**	**1.000**	0.622	0.910	0.739	**1.000**	0.143	0.250	0.402	0.779	0.530	**1.000**	**1.000**	**1.000**
	T18	www.dentists.com	**1.000**	**1.000**	**1.000**	0.634	0.810	0.711	0.228	0.787	0.354	0.315	0.557	0.402	0.863	0.863	0.863
	T19	www.drscore.com	**0.857**	**0.959**	**0.905**	0.453	0.619	0.523	0.467	0.341	0.394	0.236	0.625	0.343	0.000	0.000	0.000
	T20	www.steadyhealth.com	**1.000**	**1.000**	**1.000**	0.220	0.962	0.358	0.560	0.519	0.539	0.370	0.681	0.480	**1.000**	0.148	0.258
Jobs	T21	careers.insightintodiversity.com	**1.000**	**1.000**	**1.000**	0.531	0.850	0.654	0.429	0.300	0.353	0.095	0.567	0.162	0.900	0.900	0.900
	T22	www.4jobs.com	**0.776**	**1.000**	**0.874**	0.241	0.944	0.384	0.000	0.000	0.000	0.254	0.794	0.385	0.500	0.042	0.077
	T23	www.6figurejobs.com	**0.975**	**0.979**	**0.977**	0.669	0.951	0.785	0.068	0.141	0.092	0.622	0.914	0.740	0.000	0.000	0.000
	T24	www.careerbuilder.com	**1.000**	**1.000**	**1.000**	0.604	0.819	0.695	0.639	0.125	0.209	0.699	0.889	0.783	0.183	0.011	0.022

Table 2 (continued)

	T25	www.jobofmine.com	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.062	0.923	0.116	0.501	0.892	0.641	0.923	0.923	0.923
Movies	T26	www.albaniam.com	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000	0.655	0.619	0.636	**1.000**	**1.000**	**1.000**
	T27	www.allmovie.com	**1.000**	**1.000**	**1.000**	0.163	0.735	0.267	0.027	0.144	0.045	0.192	0.750	0.306	0.000	0.000	0.000
	T28	www.citwf.com	**1.000**	**1.000**	**1.000**	0.417	0.879	0.566	0.811	0.429	0.561	0.549	0.604	0.575	0.967	0.967	0.967
	T29	www.disneymovieslist.com	**1.000**	**0.981**	**0.991**	0.673	0.844	0.749	0.117	0.242	0.158	0.389	0.804	0.524	0.000	0.000	0.000
	T30	www.imdb.com	**0.964**	**0.799**	**0.874**	0.596	0.399	0.478	0.296	0.275	0.285	0.497	0.397	0.441	0.750	0.009	0.017
	T31	www.soulfilms.com	**0.900**	0.600	**0.720**	0.317	**0.649**	0.426	0.098	0.348	0.153	0.419	0.679	0.518	0.000	0.000	0.000
Estate	T32	realestate.yahoo.com	**0.993**	**0.962**	**0.977**	0.602	0.648	0.624	0.182	0.350	0.239	0.349	0.622	0.447	0.667	0.024	0.045
	T33	www.haart.co.uk	**1.000**	**1.000**	**1.000**	0.720	0.810	0.762	0.159	0.311	0.210	0.366	0.718	0.485	0.000	0.000	0.000
	T34	www.homes.com	**1.000**	**1.000**	**1.000**	0.678	0.472	0.557	0.678	0.072	0.130	0.395	0.392	0.394	0.000	0.000	0.000
	T35	www.remax.com	**0.938**	**0.968**	**0.953**	0.635	0.803	0.709	0.654	0.111	0.190	0.446	0.654	0.530	0.000	0.000	0.000
	T36	www.trulia.com	**0.990**	**0.981**	**0.985**	0.585	0.765	0.663	0.398	0.384	0.391	0.410	0.577	0.480	0.000	0.000	0.000
Sports	T37	baseball.playerprofiles.com	**1.000**	**1.000**	**1.000**	0.383	0.915	0.540	0.005	0.111	0.010	0.296	0.759	0.426	**1.000**	**1.000**	**1.000**
	T38	en.uefa.com	**1.000**	**1.000**	**1.000**	0.519	0.779	0.623	0.000	0.000	0.000	0.191	0.725	0.302	0.000	0.000	0.000
	T39	www.atpworldtour.com	**0.866**	0.881	**0.874**	0.650	**0.884**	0.749	0.236	0.351	0.282	0.316	0.558	0.403	0.000	0.000	0.000
	T40	www.nfl.com	**0.999**	**0.999**	**0.999**	0.746	0.984	0.849	0.050	0.241	0.083	0.376	0.841	0.519	0.979	0.904	0.940
Average			0.959	0.978	0.962	0.588	0.840	0.660	0.345	0.341	0.260	0.373	0.655	0.454	0.499	0.384	0.396

The bold numbers present the highest performance metrics in the comparison methods

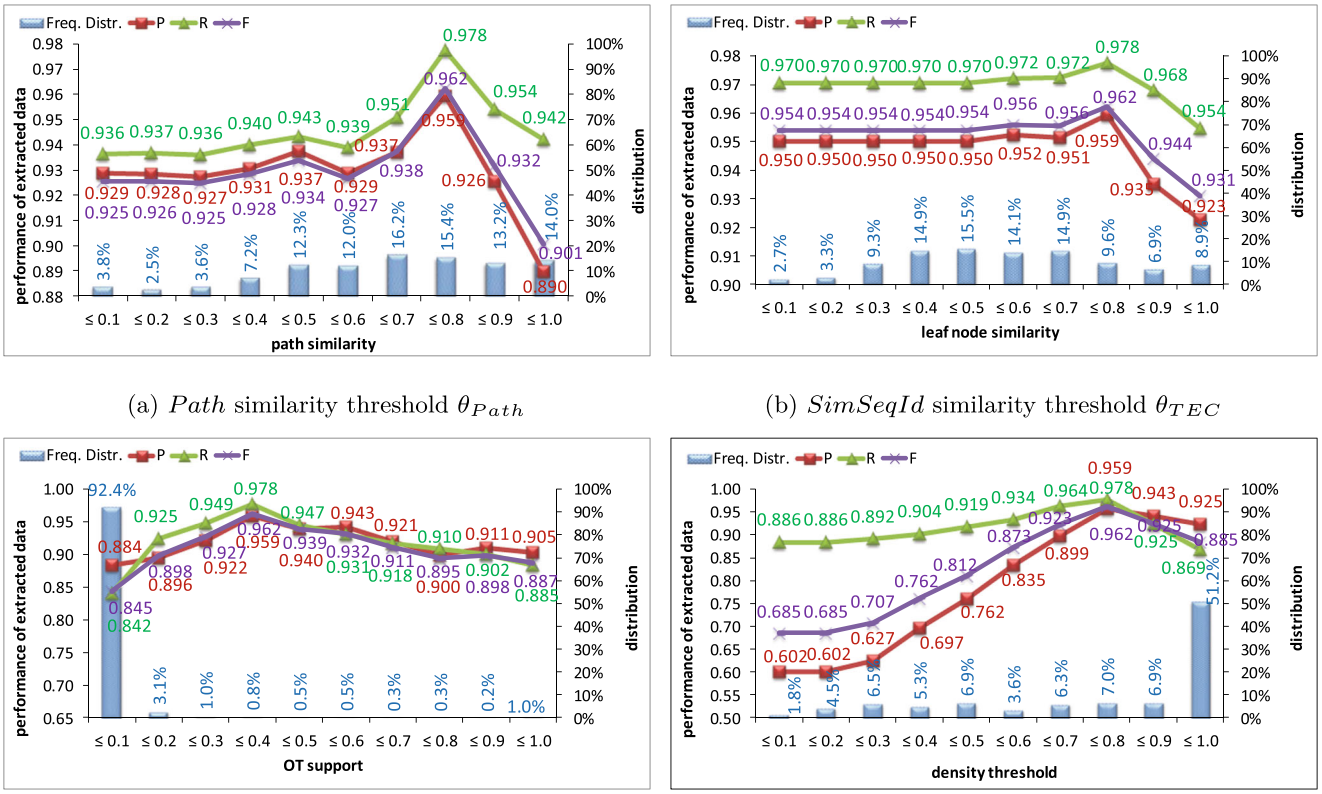
Table 3 Performance comparison on data columns of recordsets

Category	ID	DCADE			UWIDE			TEX			RoadRunner		
		P_S	R_S	F_S	P_S	R_S	F_S	P_S	R_S	F_S	P_S	R_S	F
ExAlg	E01	**1.000**	**0.951**	**0.975**	0.111	0.016	0.028	**1.000**	0.611	0.759	0.593	0.815	0.686
	E02	**1.000**	**1.000**	**1.000**	0.996	0.498	0.664	**1.000**	0.267	0.421	**1.000**	**1.000**	**1.000**
	E03	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000	**1.000**	0.986	0.993	**1.000**	**1.000**	**1.000**
	E07	0.996	0.159	0.274	**0.997**	0.797	0.886	0.995	0.411	0.582	**0.997**	**1.000**	**0.999**
Books	T01	**1.000**	**0.983**	**0.992**	0.002	0.007	0.000	0.336	0.482	0.396	0.000	0.000	0.000
	T02	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000	0.440	0.402	0.420	0.875	0.875	0.875
	T03	**1.000**	**0.952**	**0.976**	0.000	0.000	0.000	0.623	0.462	0.531	0.000	0.000	0.000
	T04	**1.000**	**0.994**	**0.997**	0.019	0.171	0.034	0.377	0.535	0.442	0.000	0.000	0.000
	T05	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000	0.407	0.530	0.460	0.000	0.000	0.000
Cars	T06	**0.871**	**0.781**	**0.823**	0.000	0.000	0.000	0.016	0.267	0.030	0.000	0.000	0.000
	T07	**1.000**	**1.000**	**1.000**	0.002	0.009	0.003	0.897	0.471	0.618	**1.000**	**1.000**	**1.000**
	T08	**1.000**	**1.000**	**1.000**	0.111	0.500	0.182	**1.000**	0.872	0.932	0.000	0.000	0.000
Events	T11	**1.000**	**1.000**	**1.000**	0.013	0.127	0.024	0.953	0.151	0.261	**1.000**	**1.000**	**1.000**
	T12	**1.000**	**1.000**	**1.000**	0.000	0.001	0.000	0.173	0.044	0.070	**1.000**	**1.000**	**1.000**
Doctors	T16	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000	0.826	0.870	0.847	0.000	0.000	0.000
	T17	**1.000**	**1.000**	**1.000**	0.018	0.434	0.035	**1.000**	0.605	0.754	**1.000**	**1.000**	**1.000**
	T19	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000	**1.000**	0.758	0.862	0.000	0.000	0.000
	T20	0.750	**1.000**	0.857	0.143	0.667	0.236	**0.985**	0.886	**0.933**	0.000	0.000	0.000
Jobs	T22	**1.000**	**0.997**	**0.999**	0.000	0.000	0.000	0.550	0.557	0.553	0.000	0.000	0.000
	T23	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000	0.115	0.678	0.197	0.000	0.000	0.000
Movies	T27	**1.000**	**1.000**	**1.000**	0.007	0.069	0.013	0.343	0.979	0.508	0.000	0.000	0.000
	T28	**1.000**	**1.000**	**1.000**	0.013	0.216	0.025	**1.000**	0.124	0.221	**1.000**	**1.000**	**1.000**
	T30	**1.000**	**0.903**	**0.949**	0.030	0.061	0.040	0.297	0.840	0.439	0.000	0.000	0.000
	T31	**1.000**	**1.000**	**1.000**	0.000	0.000	0.000	**1.000**	0.888	0.941	0.000	0.000	0.000
Estate	T32	**0.990**	**0.990**	**0.990**	0.107	0.125	0.115	0.410	**1.000**	0.582	0.000	0.000	0.000
	T34	**1.000**	**1.000**	**1.000**	0.000	0.034	0.000	0.726	0.705	0.715	0.000	0.000	0.000
	T35	0.697	**0.637**	**0.666**	0.007	0.270	0.014	**1.000**	0.193	0.324	0.000	0.000	0.000
	T36	**1.000**	**0.984**	**0.992**	0.035	0.341	0.063	0.327	0.827	0.469	0.000	0.000	0.000
Sports	T39	**1.000**	0.474	0.643	0.000	0.000	0.000	0.733	**0.616**	**0.669**	0.000	0.000	0.000
Average		0.976	0.924	0.936	0.090	0.150	0.081	0.673	0.587	0.549	0.326	0.334	0.330

The bold numbers present the highest performance metrics in the comparison methods

Among these five approaches, TEX is the only one that considers word tokens. Thus, TEX usually extracted more data columns. TEX outputs each data column in a file and ignores template columns. Moreover, TEX also skips null data cells as shown in the blank lines of the third table of Fig. 9, we therefore need to compare each line in the file with each cell in a data column of the ground truth. Similarly, we compare a basic-type element of RR and UWIDE, or a cell in *TableA* of DCADE and DCA to a ground truth cell based on character similarity using (2). If the similarity is greater than the given threshold, we consider the cell to be correctly extracted; otherwise, it is incorrect.

Since the precision of an extracted data column depends on the pair-wise cell comparison, we tuned the similarity threshold for RoadRunner and TEX to find the best performance. Figure 10a shows that RoadRunner has higher precision than recall and F1, which drops in line with the increasing threshold. On the contrary Fig. 10b has higher recall than precision as TEX extracts too many files. The performance also drops in line with the increasing threshold. To minimize false positive data extraction, we set 0.5 for RoadRunner and TEX. We set pair-wise character similarities threshold 0.7 for DCA and UWIDE and 0.85 for DCADE.



(a) *Path* similarity threshold θ_{Path}

(b) *SimSeqId* similarity threshold θ_{TEC}

(c) Support threshold θ_{OT}

(d) Section density threshold θ_{Den}

Fig. 11 Sensitivity analysis for parameter tuning

4.2 Performance comparison

Table 2 shows the performances comparison on non-recordsets, i.e. *TableA* output.[4] In terms of data column extraction, DCADE presents the best performance for all categories with $F = 0.962$, followed by DCA, TEX, RoadRunner, and UWIDE with $F = 0.660, 0.454, 0.396$ and 0.260, respectively. This is mainly because of the difference between the number of output data columns and the ground truth as shown in the extraction output comparison on Table 5 in the appendix, where UWIDE and RoadRunner output 111 and 7 data columns respectively, while the ground truth is only 49 on average. We can also examine the density of UWIDE (0.580) and RoadRunner (0.589) to know that the outputs are not as good as that of DCADE (0.917), DCA (0.900) and TEX (0.880). Note that RoadRunner deal with only 27 websites. Thus, RoadRunner is not a robust method. However, RoadRunner performs better than DCA, TEX, and UWIDE for several websites when webpages are well-formed such as E01, E02, E03, etc. (as shown in Table 2 with cyan color).

As for the evaluation in recordsets, Table 3 shows the performance of four approaches (DCA is excluded since it does not support recordset extraction). DCADE showed the best performance with $F = 0.936$ followed by TEX, RoadRunner, and UWIDE with $F = 0.549$, 0.330, and 0.081 respectively. DCADE outperformed all other approaches in every category except for ExAlg and Sports. The detail output of each approach is shown in Table 5. Due to space limitation, we could not show the density in the recordset output for each website and only report overall density in recordsets. Two better approaches are DCADE and TEX, which has an average density of 0.960 and 0.827, respectively. Two poor methods are UWIDE and RoadRunner, which has an average density of 0.621 and 0.343, respectively. However, RoadRunner has the best

[4]The detail extraction output of each approach can be found in the Appendix section.

average performance in ExAlg ($F = 0.921$) category with cyan color, where the web contents are formatted with table tags.

Although DCADE outperforms other approaches, the performance is not good on some websites (i.e., E07, T30, T31, and T39). There are three major reasons. First, when a data region contains records with skew length, DCADE may fail to detect them because the NSD is greater than 0.5. Second, false positive *MT*s or *ST*s (because of the same data values across records) may divide a data region into several parts. For example, the same data values in a data region (such as record numbers) are often detected as *MT*s or *ST*s, leading to fragmented data regions. Third, DCADE fails to detect recordsets that are mixed in one segment without sub-landmarks like *ST*.

4.3 Sensitivity analysis

We conduct the following sensitivity analysis to evaluate the robustness of the proposed algorithm. First, Fig. 11a shows the performance changes with the varying path similarity threshold (θ_{Path}) and the path similarity distribution. Path similarity is used in (2) to determine if two leaf nodes have the same *PathId*. If θ_{Path} is set too high, leaf nodes with the same text content but slightly different paths could not be generalized in the same column. If the threshold is set too low, there would be no distinction. The F1 performance varies between 0.901 and 0.962. The highest F1 performance was achieved when path similarity was 0.7 ($0.7 < \theta_{Path} \leq 0.8$).

Second, the performance (0.931∼0.962) with varying $SimSeqId$ threshold (θ_{TEC}) is shown in Fig. 11b. This parameter is used in (3) to determine if two leaf nodes have the same $SimSeqId$ and can form a template equivalence class. Similar to the path threshold, θ_{TEC} should not be set too high or too low. The best performance was achieved with $\theta_{TEC} = 0.7$ ($0.7 < \theta_{TEC} \leq 0.8$) in our experiment.

Third, the performance (0.845∼0.962) with varying support threshold θ_{OT} is shown in Fig. 11c. This parameter is used in (4) to determine whether a *CECId* is considered frequent. If the threshold is set too high, there will be fewer frequent patterns to split segments into subsegments. DCADE achieves the best performance when $\theta_{OT} = 0.3$ ($0.3 < \theta_{OT} \leq 0.4$). Most *CECId*s (92.4%) have the support less than 0.1, indicating many data nodes.

Finally, we show the performance with varying density threshold θ_{Den} for section merging and density distribution in Fig. 11d. We can see 51.2% section density higher than 0.9. Different from above, we conduct merging stops when section density is smaller than θ_{Den}. If θ_{Den} is too small, no merging is conducted and the F1 performance is only 0.685. The best performance ($F = 0.962$) was achieved when $\theta_{Den} = 0.7$ ($0.7 < \theta_{Den} \leq 0.8$).

5 Related work and comparison

The World Wide Web is the hugest data repository of information about almost any topic. However, since the data is not formatted for machine access, automatic web information extractions (IE) are important for data reuse, integration, and analysis. In this paper, we focus on the deep web which contains consolidated data from the searchable web database.

As reported in [5, 11], various IE tasks can be defined based on their input and output. Some IE tasks require annotated input pages for extraction rule induction (called supervised approaches), and some work on annotation-free input pages with or without user intervention. We call these semi-supervised and unsupervised approaches, respectively. Many IE tasks operate on list pages that contain multiple data records in a web page, while some focus on extraction from singleton pages, which contain detailed information of an item. The former usually takes a single web page as input for data extraction, such as TEGRA [7], SYNTHIA [19], AutoRM [24], Dual-TLBO [33] and MiBAT [27], while the later usually receive multiple web pages as input (e.g., RoadRunner, FivaTech [16], TEX, DCA). There are also IE systems that work with both single and multiple web pages, such as STEM [30].

In terms of output, some IE tasks are designed to extract record lists from list pages, such as [7, 17, 29, 30], while some tasks aim to induce full schema from multiple pages such as ExAlg [1], WEIR [3], RoadRunner [8], FivaTech [16], TEX [25], and DCA [31]. Essentially, multiple web pages give chances for full schema induction and can be considered a total solution if both list and singleton pages can be processed.

RoadRunner [8] is the pioneer work that generates a wrapper with full schema by aligning one page with the previous wrapper iteratively based on HTML tags. Since RoadRunner looks for paired HTML tags during string alignment, it fails to generalize well on mal-formatted web pages. Meanwhile, when input pages contain decorative tags or similar patterns for different structures, RoadRunner may fail to infer the schema, leading to the extraction of few data columns.

ExAlg detects and constructs [1] templates from large and frequent equivalence classes of tokens (either words or

HTML tags) that have the same occurrence vectors across input pages. The major problem with ExAlg is its inability to deal with inconsistent data sequences. Another issue involves how to choose the right pattern from a large number of large and frequent equivalence classes due to optional templates.

By contrast, TEX [25] and Trinity [26] find shared token sequences among all web pages and considers them as separators for data column extraction. However, TEX would extract incomplete information when all records share the same tokens.

UWIDE [6] is composed of two subsystems, i.e. page-level schema induction and schema verification. We only use the first part for the page-level schema induction, which includes recordset region detection, record boundary segmentation, data alignment and schema generation. (In contrast, our work segments webpages using *MT* templates first, followed by frequent pattern mining, repetitive pattern alignment, recordset region detection, and record boundary segmentation.) UWIDE encodes leaf node sequences based on HTML tags and text content and mines repetitive patterns for recordset based on maximal patterns. Since it segments webpages locally, there may exist false positive and false negative templates or recordsets.

DCA [31] focuses on singletton pages that contain detail description of single items and features the technique of multi-order attribute value pair extraction. Thus, DCA could not deal with list pages that contain recordsets or singleton pages that contain small data regions (e.g. Fig. 1). In addition, the alignment performance is limited because of the fixed encoding scheme.

Since this paper focuses on full schema data extraction from deep Web, we therefore exclude IE systems that do not focus on deep web data extraction, i.e. IE systems that extract only recordsets without full schema. For example, TEGRA [7], SYNTHIA [19], AutoRM [24], and Dual-TLBO [33] process one page at a time and needs extra step to align data extracted from multiple pages. Lu et al. [17] and DIADEM [29] extracts only recordsets from list pages and do not output full schema data. Therefore, we only compare with RoadRunner, TEX, UWIDE, and DCA.

In summary, Roadrunner uses iterative approaches to align multiple input sequences, whereas ExAlg, TEX, and UWIDE mine common patterns as landmarks for dividing input sequences into segments. However, existing approaches fail to work well because they are designed to align processing units with the same text contents or path. Although DCA aligns processing units with the same text contents and equal path with a fixed encoding scheme, it is hard to align various data nodes in the same column or differentiate similar data nodes into different columns. By contrast, the proposed DCADE algorithm is able to divide the input pages into segments using *MT* templates (based on *CECId*), and recursively mine *RT*, *RD*, and *ST* patterns for sub-segment division using *TECId*. The different encoding schemes extend to column rearrangement such that disjunctive column could be merged if they share common *TECId* code at a higher encoding level.

6 Conclusions

In this paper, we present a novel algorithm for web data extraction and compare the performance with DCA, UWIDE, TEX and RoadRunner, on the full pages for list and singleton pages. Compared with full web page data extraction like TEX and RoadRunner, where performance is only evaluated based on selected data, the proposed method DCADE is more robust and efficient. We conducted experiments on 49 websites from TEX and ExAlg datasets. For non-recordset data extraction, DCADE ($F_D = 0.962$) outperformed DCA ($F_D = 0.660$), TEX ($F_D = 0.454$), RoadRunner ($F_D = 0.396$), and UWIDE ($F_D = 0.260$) in column-wise evaluation. In cell-wise evaluations for recordset data extraction, DCADE ($F_S = 0.936$) outperformed TEX ($F_S = 0.549$), RoadRunner ($F_S = 0.330$), and UWIDE ($F_S = 0.081$).

The big improvement of the proposed algorithm co-mes from the following design features. First, DCADE defines *CECId* to identify *MT* for divide-and-conquer alignment. Second, DCADE systematically differentiates two types of patterns from *CECId* (i.e., *RT* and *ST*) and two other patterns from *TECId* (i.e., *RD* and *SD*) in each segment such that frequent patterns of the same type are consistent. Third, DCADE designed a postponed alignment to handle documents that do not contain a landmark.

For future work, although the performance for full schema data extraction has been improved, the extraction performance on recordset is not good when there are cascade tables or nested sets. In addition, efficient extraction on testing pages and verification of the generated scheme can complete the task for wrapper maintenance.

Acknowledgements The research is supported by Ministry of Science and Technology Taiwan under Grant MOST105-2628-E-008-004-MY2.

Appendix

The magenta numbers in brackets represent the gap with the ground truth answer.

Table 4 Extraction output comparison on *TableA*

Category	ID	DCADE				DCA		UWIDE		TEX		RoadRunner	
		#Column	#Template	#Data	Density	#Data	Density	#Data	Density	#Data	Density	#Data	Density
ExAlg	E01	(+1) 142	(+1) 136	5	1.000	(+7) 12	0.972	(+18) 23	0.569	(+14) 16	1.000	5	1.000
	E02	36	28	7	1.000	(+17) 24	1.000	(−6) 1	1.000	(+15) 17	0.947	(+2) 9	0.894
	E03	229	223	5	1.000	(+162) 167	0.877	(+302) 307	0.483	(+6) 11	1.000	5	1.000
	E04	56	39	17	0.997	17	0.991	(−4) 13	1.000	(+2) 16	1.000	(+2) 19	0.997
	E05	142	106	36	0.938	(+6) 42	0.955	(−26) 10	1.000	(+19) 35	0.957	36	0.938
	E06	141	102	39	0.889	(+14) 53	0.749	(+60) 99	0.366	(+46) 51	0.803	(−37) 2	1.000
	E07	(+9) 365	352	(+8) 11	(−0.115) 0.885	(+21) 24	0.996	3	1.000	(+51) 15	0.973	3	1.000
	E08	(+1) 391	317	(+1) 74	(−0.013) 0.944	(+7) 80	0.959	(+30) 103	0.518	(+29) 61	0.852	(−73) 0	0.000
	E09	24	20	4	(−0.004) 0.996	(+2) 6	0.933	(+716) 720	0.268	(+10) 9	0.922	(−4) 0	0.000
Books	T01	(−1) 302	255	(−1) 45	(+0.009) 0.899	(+6) 52	0.851	(+54) 100	0.484	(+6) 44	0.977	(−46) 0	0.000
	T02	216	191	24	0.844	(+1) 25	0.831	(+53) 77	0.376	(+13) 27	0.784	(−1) 23	0.922
	T03	(+13) 378	(+7) 307	(+6) 69	(−0.020) 0.929	(+4) 67	0.965	(−63) 0	0.000	(+139) 159	0.841	(−58) 5	1.000
	T04	146	114	31	(+0.004) 0.857	(+6) 37	0.910	(+242) 273	0.287	(+20) 33	0.726	(−31) 0	0.000
	T05	301	287	13	0.860	(+10) 23	0.884	(+80) 93	0.546	(+98) 88	0.669	(−8) 5	1.000
Cars	T06	(−3) 493	(−4) 315	(+1) 176	(+0.001) 0.910	(+3) 178	0.888	(+157) 332	0.653	(+10) 117	0.954	(−175) 0	0.000
	T07	391	(+1) 322	(−1) 68	(−0.003) 0.954	(+67) 136	0.893	(+27) 96	0.769	(+50) 91	0.873	(−55) 14	1.000
	T08	196	145	50	0.877	(+11) 61	0.856	(−11) 39	0.790	(+5) 41	0.880	(−50) 0	0.000
	T09	580	517	63	(−0.015) 0.917	(+4) 67	0.925	(+51) 114	0.685	(+50) 85	0.871	(−63) 0	0.000
	T10	(+1) 310	228	(+1) 82	(−0.011) 0.967	(+48) 129	0.918	(+48) 129	0.772	(+29) 79	0.893	(−81) 0	0.000
Events	T11	82	52	27	0.889	(+5) 32	0.917	(−22) 5	0.800	(+6) 27	0.927	(−1) 26	0.874
	T12	115	88	26	0.900	(+17) 43	0.825	(+7) 33	0.876	(+8) 31	0.908	26	0.896
	T13	50	37	13	1.000	13	1.000	(−11) 2	1.000	(+2) 15	1.000	13	1.000
	T14	1	0	1	0.974	(+1) 2	0.983	(−1) 0	0.000	(+6) 4	1.000	1	1.000
	T15	46	39	7	1.000	(+10) 17	0.816	(−4) 3	1.000	(+5) 8	0.883	(−5) 2	1.000
Doctors	T16	(+8) 559	(+4) 537	(+4) 21	(−0.100) 0.749	(+1) 18	0.832	(+32) 49	0.506	(+7) 20	0.815	(−14) 3	1.000
	T17	270	255	14	0.922	(+7) 21	0.890	(−12) 2	1.000	(+10) 22	0.747	14	0.917
	T18	61	51	10	0.960	(+4) 14	0.862	(+25) 35	0.543	(+8) 13	0.936	10	0.960
	T19	122	95	25	0.926	(+10) 35	0.873	(−5) 20	0.775	(+39) 56	0.807	(−25) 0	0.000
	T20	332	304	27	0.956	(+91) 118	0.959	(−2) 25	1.000	(+27) 42	0.971	(−23) 4	1.000
Jobs	T21	175	165	10	1.000	(+6) 16	0.938	(−3) 7	1.000	(+42) 32	0.613	10	1.000
	T22	(+8) 197	161	(+8) 32	(−0.193) 0.712	(+72) 96	0.899	(−24) 0	0.000	(+55) 72	0.922	(−22) 2	1.000
	T23	(+1) 653	613	(+1) 38	(−0.024) 0.950	(+16) 53	0.847	(+40) 77	0.454	(+16) 39	0.976	(−37) 0	0.000
	T24	221	189	32	0.979	(+12) 44	0.870	(−24) 8	0.713	(+6) 31	0.953	(−30) 2	1.000

Table 4 (continued)

	T25	98	85	13	0.992	13	0.992	(+180) 193	0.261	(+9) 14	0.979	13	0.992
Movies	T26	93	75	18	0.674	18	0.674	(−18) 0	0.000	(+1) 17	1.000	18	0.674
	T27	145	117	22	0.952	(+84) 106	0.865	(+162) 184	0.371	(+62) 56	0.830	(−22) 0	0.000
	T28	79	61	16	0.835	(+19) 35	0.757	(−7) 9	0.789	(+8) 24	0.696	16	0.815
	T29	132	105	27	(−0.019) 0.967	(+8) 35	0.804	(+39) 66	0.389	(+26) 42	0.773	(−27) 0	0.000
	T30	(+11) 1,100	(+1) 734	(+9) 360	(−0.011) 0.816	(−111) 240	0.917	(+34) 385	0.765	(+86) 208	0.953	(−347) 4	1.000
	T31	259	229	29	0.937	(+32) 61	0.970	(+87) 116	0.526	(+16) 33	0.938	(−29) 0	0.000
Estate	T32	449	354	85	0.889	(+9) 94	0.912	(+115) 200	0.551	(+82) 139	0.787	(−82) 3	1.000
	T33	228	181	47	0.930	(+7) 54	0.898	(+67) 114	0.434	(+44) 70	0.811	(−47) 0	0.000
	T34	302	209	92	0.648	(−27) 65	0.870	(−81) 11	0.491	(+10) 79	0.723	(−92) 0	0.000
	T35	(+5) 276	(+3) 210	(+2) 61	(−0.013) 0.891	(+18) 77	0.895	(−48) 11	0.736	(+23) 57	0.899	(−59) 0	0.000
	T36	(+5) 959	(+9) 576	(−4) 379	(−0.002) 0.951	(+138) 521	0.930	(+26) 409	0.720	(+126) 410	0.899	(−383) 0	0.000
Sports	T37	53	35	18	0.972	(+25) 43	0.966	(+540) 558	0.281	(+25) 30	0.809	18	0.972
	T38	301	282	19	0.991	(+11) 30	0.883	(−19) 0	0.000	(+54) 58	0.782	(−19) 0	0.000
	T39	(+3) 817	(+1) 744	(+2) 72	(−0.018) 0.902	(+25) 95	0.928	(+41) 111	0.574	(+46) 69	0.958	(−70) 0	0.000
	T40	651	599	52	0.986	(+17) 69	0.964	(+203) 255	0.326	(+59) 78	0.883	(−4) 48	0.985
Average		279	228	49	0.917	67	0.900	111	0.580	57	0.880	7	0.589

Table 5 Extraction output comparison on recordsets

Category	ID	DCADE				UWIDE			TEX			RoadRunner		
		#Set	#Template	#Data	*MaxRow*	#Set	#Data	*MaxRow*	#Set	#Data	*MaxRow*	#Set	#Data	*MaxRow*
ExAlg	E01	1	3	8	54	1	(−7) 1	(+9) 63	1	(−1) 7	(−21) 33	1	(+3) 11	54
	E02	1	0	2	501	1	(−1) 1	501	1	2	(−367) 134	1	2	501
	E03	1	1	2	2,807	1	(+6) 8	(−2,750) 57	1	2	(−38) 2,769	1	2	2,807
	E07	(+1) 2	0	5	(+15) 567	1	(−1) 4	552	1	5	(−324) 228	1	5	552
Books	T01	2	2	10	96	2	(+1) 11	(+59) 155	2	(−7) 3	(−66) 30	(−2) 0	(−10) 0	(−96) 0
	T02	1	2	8	47	1	(−2) 6	(+19) 66	1	(−6) 2	(−17) 30	1	8	47
	T03	2	(−3) 4	(+3) 11	478	(−2) 0	(−8) 0	(−478) 0	2	(−2) 6	(−275) 203	(−2) 0	(−8) 0	(−478) 0
	T04	1	1	2	85	1	(+16) 18	(+2) 87	1	(−1) 1	(−63) 22	(−1) 0	(−2) 0	(−85) 0
	T05	1	0	3	39	(+1) 2	(−1) 2	(+112) 151	1	(−1) 2	(−9) 30	(−1) 0	(−3) 0	(−39) 0
Cars	T06	2	2	2	(−19) 188	(−1) 1	(+5) 7	(+63) 270	(−1) 1	(−1) 1	(−177) 30	(−2) 0	(−2) 0	(−207) 0
	T07	1	1	8	162	(+3) 4	(+14) 22	(+48) 210	1	(−4) 4	(−95) 67	1	(−1) 7	162
	T08	1	1	2	234	(+2) 3	(+16) 18	(+60) 294	1	2	(−30) 204	(−1) 0	(−2) 0	(−234) 0
Events	T11	3	1	6	302	(−1) 2	(+97) 103	(−46) 256	3	6	(−236) 66	3	6	302
	T12	1	1	2	842	1	(+33) 35	(−639) 203	1	2	(−812) 30	1	2	842
Doctors	T16	1	0	3	115	(−1) 0	(−3) 0	(−115) 0	1	3	(+6) 121	(−1) 0	(−3) 0	(−115) 0
	T17	1	1	2	76	1	(+14) 16	(+159) 235	1	2	(−30) 46	1	2	76
	T19	2	2	2	76	2	(+2) 4	(+177) 253	2	2	(−26) 50	(−2) 0	(−2) 0	(−76) 0
	T20	1	0	4	600	(+3) 4	(+10) 14	600	1	(−1) 3	(−60) 540	(−1) 0	(−4) 0	(−600) 0
Jobs	T22	4	3	18	(−3) 205	(−4) 0	(−18) 0	(−208) 0	4	(−6) 12	(−108) 100	(−4) 0	(−18) 0	(−208) 0
	T23	2	2	2	124	(−1) 1	(+2) 4	(−64) 60	2	2	(−94) 30	(−2) 0	(−2) 0	(−124) 0
Movies	T27	6	1	12	404	(−4) 2	(+20) 32	(+122) 526	6	(−1) 11	(−60) 344	(−6) 0	(−12) 0	(−404) 0
	T28	2	2	3	241	2	(+17) 20	(+17) 258	2	(−1) 2	(−211) 30	2	3	241
	T30	(+1) 6	3	14	350	(−3) 2	(−6) 8	(+96) 446	5	(−1) 13	(+66) 416	(−5) 0	(−14) 0	(−350) 0
	T31	1	0	2	535	1	(+1) 3	(−425) 110	1	2	(−60) 475	(−1) 0	(−2) 0	(−535) 0
Estate	T32	10	1	30	407	(−7) 3	(−6) 24	(−237) 170	(−1) 9	(−10) 20	(+78) 485	(−10) 0	(−30) 0	(−407) 0
	T33	–	–	–	–	–	–	–	–	–	–	–	–	–
	T34	1	1	2	117	1	(+248) 250	(+216) 333	1	(−1) 1	(−104) 13	(−1) 0	(−2) 0	(−117) 0
	T35	5	5	6	290	(−1) 4	(+36) 42	(+198) 488	(−1) 4	(−2) 4	(+5) 295	(−5) 0	(−6) 0	(−290) 0
	T36	4	1	16	150	4	(+75) 91	(+300) 450	4	(−5) 11	(+229) 379	(−4) 0	(−16) 0	(−150) 0
Sports	T39	1	1	3	(−27) 32	1	(+7) 10	(+1) 60	1	1 (−2) 1	(−3) 56	(−1) 0	(−3) 0	(−59) 0
Average		3	2	7	349	2	26	237	3	5	251	1	2	193

References

1. Arasu A, Garcia-Molina H (2003) Extracting structured data from web pages. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pp 337–348
2. Bing L, Lam W, Wong TL (2013) Wikipedia entity expansion and attribute extraction from the web using semi-supervised learning. In: Proceedings of the sixth ACM international conference on Web search and data mining, pp 567–576
3. Bronzi M, Crescenzi V, Merialdo P, Papotti P (2013) Extraction and integration of partially overlapping web sources. VLDB J 6(10):805–816
4. Carlson A, Betteridge J, Wang RC, Hruschka R, Mitchell TM (2010) Coupled semi-supervised learning for information extraction. In: Proceedings of the third ACM international conference on Web search and data mining, pp 101–110
5. Chang CH, Kayed M, Girgis MR, Shaalan KF (2006) A survey of web information extraction systems. IEEE Trans Knowl Data Eng 18(10):1411–1428
6. Chang CH, Chen TS, Chen MC, Ding JL (2016) Efficient page-level data extraction via schema induction and verification. In: Proceedings of the Pacific-Asia conference on knowledge discovery and data mining, pp 478–490
7. Chu X, He Y, Chakrabarti K, Ganjam K (2015) Tegra: table extraction by global record alignment. In: Proceedings of the 2015 ACM SIGMOD international conference on management of data, pp 1713–1728
8. Crescenzi V, Mecca G (2005) Automatic information extraction from large websites. Journal of the ACM (JACM) 51(5):731–779
9. Crescenzi V, Merialdo P, Alfred DQ (2013) Alfred: crowd assisted data extraction. In: Proceedings of the 22nd international conference on World Wide Web, pp 297–300
10. Dhillon PS, Sellamanickam S, Selvaraj SK (2011) Semi-supervised multi-task learning of structured prediction models for web information extraction. In: Proceedings of the 20th ACM international conference on information and knowledge management, pp 957–966
11. Ferrara E, De Meo P, Fiumara G, Baumgartner R (2014) Web data extraction, applications and techniques: a survey. Knowl-Based Syst 70:301–323
12. Furche T, Gottlob G, Grasso G, Schallhart C, Sellers A (2013) OXPAth: a language for scalable data extraction, automation, and crawling on the deep web. The International Journal on Very Large Data Bases 22(1):47–72
13. Gulhane P, Madaan A, Mehta R, Ramamirtham J, Rastogi R, Satpal S, Sengamedu SH, Tengli A, Tiwari C (2011) Web-scale information extraction with vertex. In: Proceedings of the IEEE 27th international conference on data engineering, pp 1209–1220
14. Gupta R, Sarawagi S (2011) Joint training for open-domain extraction on the web: exploiting overlap when supervision is limited. In: Proceedings of the fourth ACM international conference on Web search and data mining, pp 217–226
15. Jiménez P, Corchuelo R (2016) On learning web information extraction rules with TANGO. Inf Syst J 66:74–103
16. Kayed M, Chang CH (2010) Fivatech: page-level web data extraction from template pages. IEEE Trans Knowl Data Eng 22(2):249–263
17. Lu Y, He H, Zhao H, Meng W, Yu C (2013) Annotating search results from web databases. IEEE Trans Knowl Data Eng 25(3):514–527
18. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–453
19. Omari A, Kimelfeld B, Yahav E, Shoham S (2016) Lossless separation of web pages into layout code and data. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1805–1814
20. Omari A, Shoham S, Yahav E (2017) Synthesis of forgiving data extractors. In: Proceedings of the tenth ACM international conference on web search and data mining, pp 385–394
21. Ortona S, Orsi G, Furche T, Buoncristiano M (2016) Joint repairs for web wrappers. In: Proceedings of IEEE 32nd international conference on data engineering, pp 1146–1157
22. Qu J, Ouyang D, Hua W, Ye Y, Zhou X (2019) Discovering correlations between sparse features in distant supervision for relation extraction. In: Proceedings of the twelfth ACM international conference on web search and data mining, pp 726–734
23. Ratner AJ, Bach SH, Ehrenberg HR, Ré C (2017) Snorkel: fast training set generation for information extraction. In: Proceedings of the 2017 ACM international conference on management of data, pp 1683–1686
24. Shi S, Liu C, Shen Y, Yuan C, Huang Y (2015) AutoRM: an effective approach for automatic Web data record mining. Knowl-Based Syst 89:314–331
25. Sleiman HA, Corchuelo R (2013) Tex: an efficient and effective unsupervised web information extractor. Knowl-Based Syst 39:109–123
26. Sleiman HA, Corchuelo R (2014) Trinity: on using trinary trees for unsupervised web data extraction. IEEE Trans Knowl Data Eng 26(6):1544–1556
27. Song X, Liu J, Cao Y, Lin CY, Hon HW (2010) Automatic extraction of web data records containing user-generated content. In: Proceedings of the 19th ACM international conference on information and knowledge management, pp 39–48
28. Su W, Wang J, Lochovsky FH, Liu Y (2012) Combining tag and value similarity for data extraction and alignment. IEEE Trans Knowl Data Eng 24(7):1186–1200
29. Tim F, Georg G, Giovanni G, Xiaonan G, Giorgio O, Christian S, Cheng W (2014) DIADEM: thousands of websites to a single database. In: Proceedings of the VLDB, vol 7, pp 1845-1856
30. Xie X, Fang Y, Zhang Z, Li L (2012) Extracting data records from web using suffix tree. In: Proceedings of the ACM SIGKDD workshop on mining data semantics, p 12
31. Yuliana OY, Chang CH (2018) A novel alignment algorithm for effective web data extraction from singleton-item pages. Appl Intell 48(11):4355–4370
32. Zhai Y, Liu B (2006) Structured data extraction from the web based on partial tree alignment. IEEE Trans Knowl Data Eng 18(12):1614–1628
33. Zhao C, Zhang R, Qi J (2018) Web page template and data separation for better maintainability. In: Proceedings of international conference on web information systems engineering, pp 439–449

Oviliani Yenty Yuliana received the B.Eng. in Computer Engineering from Institut Teknologi Sepuluh Nopember, Indonesia, 1993 and the MS in Computer Information System from Assumption University, Thailand, 2004. Currently, she is a Ph.D. student at Computer Science and Information Engineering, National Central University, Taiwan. Her research interests are Database Systems, Data Mining, and Web Data Extraction.

Chia-Hui Chang is a full Professor at National Central University, Taiwan. She obtained her Ph.D. in Computer Science and Information Engineering from National Taiwan University, Taiwan in 1999. Her research interests focus on Information Extraction, Web Intelligence, Data Mining, Machine Learning and System Integration. Dr. Chang has published more than 80 papers at refereed conferences and journals including WWW, PAKDD, TKDE, IEEE Intelligent Systems, etc. She served as area co-chairs for ACL 2017, NAACL 2018 and senior PC members for AAAI, ICTIR, PAKDD, etc. She is currently president of Taiwan Association for Artificial Intelligence (TAAI) and vice president for Association for Computational Linguistic and Chinese Language Processing (ACLCLP).